Escuela Politécnica Superior

Entorno 3D en Unreal Engine 4

Grado en Ingeniería Multimedia

Trabajo Fin de Grado

Autor:
Marina López Menárguez

Tutor/es:
Mireia Sempere Tortosa

Junio 2018

Universitat d'Alacant
Universidad de Alicante

Justificación y objetivos

Para finalizar el Grado en Ingeniería Multimedia, quería llevar a cabo un proyecto práctico; un proyecto que aprovechara tecnologías de la actualidad. Siendo los gráficos en tres dimensiones y el de la realidad virtual campos en creciente desarrollo, me decanté por realizar un trabajo donde convergieran ambos.

Decidí llevar a cabo el proyecto en Unreal Engine 4, al ser un motor ampliamente utilizado en videojuegos y entornos gráficos durante los últimos años.

El tipo de entorno por el que me decanté está inspirado en uno de mis videojuegos favoritos: *Life is Strange* –cuyo motor es Unreal Engine 3–, donde hay escenarios paisajísticos estilizados.

Así pues, mi principal objetivo era crear un entorno paisajístico en tres dimensiones con un estilo realista, con gran cantidad de detalle, para después añadirle la inmersión característica de la realidad virtual.

En definitiva, me propuse aprender a trabajar con tecnologías actuales para realizar un producto que pueda ser utilizado en el ámbito de los videojuegos y la realidad virtual de hoy en día.

Índice de contenidos

Índice de figuras

1. Estado del arte

¿Qué es la realidad virtual? Para Albert Einstein, "la realidad es meramente una ilusión, aunque una muy persistente".

Reality is merely an illusion, albeit a very persistent one.

Albert Einstein

Así pues, la realidad virtual consiste en la ilusión de inmersión en un entorno, esto es, hacer creer al cerebro del usuario que se está en otro lugar. Se trata por tanto de un entorno u objetos de apariencia real.

Para crear esta ilusión se utilizan diversas tecnologías de software y hardware.

1.1. Tecnologías de realidad virtual

Pantallas estereoscópicas

También conocidos como "dispositivos 3D", "casco/gafas de realidad virtual" o HMDs (del inglés *head-mounted displays*), estos dispositivos utilizan una combinación de múltiples imágenes, una distorsión óptica realista, y lentes especiales para producir una imagen estéreo que crea esa sensación de profundidad (3 dimensiones) y por tanto de inmersión.

El render debe hacerse al menos a 60 fps para evitar la latencia, la cual podría romper esta ilusión e incluso provocar náusea.

Durante años, uno de los mayores impedimentos para el consumismo de realidad aumentada era la inexistencia de un dispositivo económico que a su vez fuera ligero y

cómodo de llevar durante un tiempo extenso. Esto cambió cuando el equipo de Oculus VR creó las Oculus Rift.

Para crear la sensación de profundidad, las Oculus Rift crean una imagen separada para cada ojo, una ligeramente desplazada respecto a la otra, para simular paralaje: un fenómeno visual en el cual el cerebro percibe la profundidad basándose en la diferencia de la posición aparente de los objetos (debido a que los ojos están ligeramente separados uno del otro). Además, para que la ilusión sea buena, se distorsiona la imagen para emular la forma esférica del ojo mediante una distorsión de barril.

Además de Oculus Rift, hay otros HMDs: algunos funcionan sólo para ordenadores de escritorio, otros sólo para smartphones, y otros se utilizan únicamente en videoconsolas. Los hay en una variedad de estilos y rango de precios.

Hardware de captura de movimiento

Se captura la posición y orientación mediante dispositivos electrónicos específicos: unidades de medición inercial o IMU (del inglés *inertial measurement unit*). Así la aplicación actualiza la vista de la escena 3D para que coincida con éstas.

Dispositivos de entrada

Para conseguir una buena realidad virtual se requiere de mandos de control u otros nuevos dispositivos para captura de manos o del cuerpo.

El sistema perceptual humano es muy sensible al movimiento, y del mismo modo que se ha hablado de la latencia en las pantallas estereoscópicas, la latencia en estos dispositivos también supone la pérdida de inmersión, así como la posibilidad de padecer náusea.

Capturando el movimiento lo más rápidamente posible, combinando con el render actualizándose a su vez, ambos a alta frecuencia, se consigue una sensación real de inmersión.

Software de plataforma

Son necesarios diferentes tipos de software para la creación de aplicaciones en realidad aumentada, tales como:

Kits de desarrollo de software o SDKs (del inglés "software development kit")

Los *drivers* (controladores) de los dispositivos y librerías de software utilizados junto al sistema operativo. Se puede crear una aplicación únicamente con SDKs, pero normalmente los desarrolladores utilizan motores y *frameworks*.

Motores de videojuegos y frameworks

Cuando no se es un desarrollador de motor de videojuegos, en lugar de manejar directamente los SDKs se prefiere trabajar con motores como Unity3D o Unreal Engine. Las librerías de estos motores se encargan del bajo nivel (renderizados, físicas, etc). En muchos casos se permite enfocar la aplicación a múltiples plataformas sin necesidad de escribir el código más de una vez.

Navegadores web

Se utilizan tecnologías de HTML5, WebGL y JavaScript para crear las aplicaciones, haciendo que sean más rápidas de programar, así como multiplataforma, y además permitiendo el acceso a infraestructuras de la Web, como hiperlinks entre experiencias de realidad virtual, alojar contenidos en la nube, desarrollar experiencias compartidas entre usuarios e integrar datos web directamente en la aplicación.

Reproductores de vídeo

Mediante el uso de múltiples cámaras (como mínimo dos) se capturan varias vistas de una escena. Si se desea un vídeo panorámico (es decir, con una vista de trescientos sesenta grados) se necesitan varias cámaras.

La captura y producción de vídeos en realidad virtual es un campo creciente del que se han interesado numerosas empresas y proyectos de investigación.

1.2. Hardware de realidad virtual

Escritorio

Es en los ordenadores de escritorio donde la realidad virtual tiene el mayor rendimiento y fidelidad. Hay un amplio campo de visión (más de 100 grados), un refresco rápido (a partir de 90 Hz) y 6 grados de libertad.

Oculus Rift

Creadas por Oculus VR, las Oculus Rift fueron las primeras gafas de realidad virtual, además de las preferidas por muchos desarrolladores.

Se trata de un dispositivo estereoscópico con sensores de captura de movimiento de la cabeza. Se coloca sobre la cabeza dejando libres las manos, y funciona junto a un ordenador (de escritorio o portátil) con sistema operativo Mac, Linux o Windows, al que se conecta por cable.

Poseen un refresco de 90 Hz y 110 grados de campo de visión. Además, pueden combinarse con un mando de control de Xbox.

Figura 1: Oculus Rift

Fuente: https://www.oculus.com/

HTC Vive

Creadas en conjunto por HTC y Valve, son unas gafas de alta resolución, con 90 Hz de refresco y campo de visión de 110 grados. Permiten *room-scale*: se utiliza un área despejada para permitir al usuario el movimiento, que se traduce en movimiento a su vez en el entorno virtual. Esto hace que la inmersión sea mayor.

Tienen sus propios mandos de control como dispositivo de entrada.

OSVR

Open Source Virtual Reality (realidad virtual de código abierto) es una especificación abierta para software y hardware en la que colaboran varias empresas, incluyendo Sensics y Razer.

Se trata de un ecosistema donde crear HMDs y dispositivos de entrada que funcionen conjuntamente, además de aplicaciones mediante APIs comunes: se crean aplicaciones sin conocer las especificaciones hardware.

Figura 2: Gafas de Razer basadas en la especificación abierta OSVR

Fuente: www.osvr.org

Samsung Gear VR

Oculus, en colaboración con Samsung, ha desarrollado estas gafas para su uso con el móvil –las Oculus Rift requieren de un ordenador potente para funcionar, por lo que no son muy portables.

Combinan las lentes de las Oculus con una nueva tecnología de captura de movimiento de la cabeza. El móvil - que ha de ser Samsung -, con imagen de alta calidad, se sitúa en el interior del dispositivo.

Tienen un campo de visión de 96 grados y una velocidad de refresco de 60 Hz. Sólo capturan la orientación, no el movimiento, y poseen una pequeña pantalla táctil como dispositivo de entrada.

Figura 3: Samsung Gear VR

Fuente: http://www.samsung.com/global/galaxy/gear-vr/

Daydream VR

Se trata de una especificación abierta que funciona para múltiples móviles y con un controlador de 3 grados de libertad.

Google Cardboard

Desarrolladas por Google, estas económicas gafas consisten en una caja de cartón y dos lentes. Tienen un campo de visión de 90 grados y funcionan para múltiples dispositivos móviles. No se ofrecen como producto: Cardboard es más bien una especificación. Todavía se está trabajando en un dispositivo de entrada. El principal inconveniente es la alta latencia.

Figura 4: Google Cardboard originales, mostrados en Google I/O en 2014

Fuente: https://vr.google.com

FOVE

Es la nueva generación de HMDs, capaz de capturar el movimiento ya no sólo de la cabeza sino también de los ojos. Esto amplía el abanico de posibilidades para las aplicaciones de realidad virtual.

Project Morpheus

Un sistema de realidad virtual creado por Sony para su consola Play Station 4. Consiste en un confortable HMD y dos mandos de control con sensibilidad de movimiento para las manos.

Dispositivos de entrada VR

Normalmente utilizar el teclado o ratón a ciegas no ofrece una buena experiencia en compatibilidad con la realidad virtual. Al eliminar la vista del usuario del mundo real, se necesitan nuevos dispositivos de entrada que generen una mayor sensación de inmersión.

Se está experimentando en éste área, con dispositivos de entrada como:

Mandos de control

Como los de las consolas Microsoft Xbox One y Sony Play Station 4. También pueden conectarse al ordenador, por lo que funcionan tanto para aplicaciones de PC como de móviles.

Sensores de captura de movimiento de las manos

Durante los últimos años han estado disponibles dispositivos de entrada de movimiento de bajo coste, como el controlador Leap Motion. Éste utiliza una combinación de cámaras y LEDs infrarrojos para capturar el movimiento de las manos y reconocer gestos, de manera similar al Xbox Kinect.

Figura 5: Leap Motion Controler en unas Oculus Rift DK2

Fuente: https://developer.leapmotion.com

Capturadores de manos y cuerpo mediante wifi

Como el sistema STEM, de Sixense, de cuerpo entero, o Hydra de Razer. Estos sistemas combinan la captura de movimiento de las manos con botones similares a los de los mandos de control.

Oculus ha desarrollado su propio dispositivo, Oculus Touch, un par de mandos que se sostienen en cada mano y funcionan vía wifi. La cámara Rift localiza la posición de estos mandos, por lo que se puede ver una representación de las manos en el mundo virtual.

Figura 6: controladores Oculus Touch

Fuente: https://www.oculus.com/

1.3. Aplicaciones de la realidad virtual

A pesar de que la realidad virtual se utiliza desde no hace mucho tiempo, tiene numerosas aplicaciones en diferentes campos:

Videojuegos

El potencial de una gran inmersión hace que la realidad virtual sea muy utilizada en los videojuegos. La mayoría de los desarrolladores de realidad virtual independientes la aplican a este campo.

Mundos virtuales

Los mundos virtuales sociales hacen una buena combinación con la realidad virtual. Empresas como High Fidelity - creada por el fundador de *Second Life*, Philip

Rosedale -, y AltSpace VR - una nueva empresa de la Bahía de San Francisco -, están liderando este campo.

Educación

La visualización 3D se ha utilizado durante años para el aprendizaje interactivo: la inmersión de realidad virtual podría hacerlo más efectivo.

Productividad

Algunos investigadores y pequeñas compañías valoran utilizar la realidad virtual como sustituto de ordenadores de escritorio utilizados únicamente para guardar datos y organizar proyectos o tareas. Se trataría de disponer del entorno de trabajo - que incluye la información personal, contactos, proyectos de trabajo, etc. - en trescientos sesenta grados mediante realidad virtual.

Turismo

Mediante los panoramas en trescientos sesenta grados de diferentes entornos reales, se consigue la experiencia de estar en otro lugar sin necesidad de viajar.

Inmobiliaria y arquitectura

Utilizan tanto vídeo como entornos interactivos. Los vídeos son un buen modo de mostrar las propiedades existentes, mientras que los entornos interactivos se usan para la visualización de construcciones en etapas de planificación.

Eventos en vivo

La realidad virtual puede llegar a ser muy popular en conciertos, en reportajes y en otros eventos en vivo. Músicos como Paul McCartney y Jack White, entre otros, transmiten versiones en realidad virtual sus conciertos en vivo.

Navegadores Web

Mozilla es líder en cuanto a la adición de soporte para realidad virtual a su navegador, y Google no está muy lejos de crear las mismas características para Chrome. Se exploran diseños virtuales e interfaces para navegar por un universo de información en realidad virtual.

Aplicaciones empresariales

Se incluyen en este campo simulación y entrenamiento para uso militar, diagnósticos médicos, ingeniería y diseño, entre otras muchas aplicaciones.

1.4. Videojuegos con realidad virtual

Zero Latency: Singularity

Desarrollado por Raven Software y adaptado a la realidad virtual por Zero Latency, se trata de un juego cooperativo contra IA de hasta dieciséis personas, equipadas con unos cascos OSVR, auriculares con micro, una mochila con un portátil y un arma. Lo que hace la experiencia tan inmersiva es el hecho de que los jugadores se

pueden mover libremente en un gran espacio. En él, un sistema de cámaras rastrea sus movimientos.

Además, para cada jugador suena una alarma cuando se acerca demasiado a otro jugador o a un obstáculo (por ejemplo, una pared).

Figura 7: Cinco jugadores de Singularity en Zero Latency

Fuente: https://www.xataka.com

Figura 8: Videojuego Zero Latency: Singularity

Fuente: https://www.xataka.com

The Lab

Creado por Valve, este videojuego sitúa al jugador en el universo de *Portal*, en el cual se puede mover libremente (utilizando un espacio físico), así como acceder a una serie de minijuegos. Se utiliza la tecnología HTC Vive y dos controladores: uno para cada mano del jugador.

Figura 9: Videjuego The Lab

Fuente: www.virtualrealmsvr.com

Robo Recall

Este shooter en primera persona fue desarrollado por Epic Games, y utiliza la tecnología de Oculus Touch. Se trata de un juego de ordenador (disponible para Microsoft Windows), por lo que en este caso no se hace uso de un espacio físico para el movimiento del jugador por los escenarios.

En el juego se debe eliminar a los robots defectuosos que van apareciendo. Hay una gran variedad de armas, y además el jugador puede agarrar a los robots, desmontarlos, lanzarlos o incluso usarlos como arma contra otros robots.

Figura 10: Videojuego Robo Recall

Fuente: https://www.roadtovr.com

Land's End

Es un juego de aventura creado por los desarrolladores de *Monument Valley*. El jugador se sitúa en un gran entorno de espectaculares paisajes, donde debe despertar a una antigua civilicación utilizando sus poderes mentales.

Utiliza Samsung Gear VR y está disponible únicamente para Samsung Galaxy S6 junto a los cascos Gear VR.

Figura 11: Videojuego Land's End

Fuente: www.landsendgame.com

2. Objetivos

Este proyecto tiene como objetivo principal la realización de un entorno virtual de un paisaje natural con un estilo realista. Ya sea a través de la pantalla del ordenador, o mediante las gafas Oculus Rift, se pretende sumergir al observador en un lugar totalmente diferente.

En el desarrollo de este proyecto se destacan los siguientes objetivos:

- Modelado y texturizado de elementos del entorno
- Creación de niveles de detalle
- Animación de viento en algunos elementos
- Animación de pájaros sobrevolando el cielo
- Creación del terreno
- Colocación de elementos sobre el terreno para conformar el entorno
- Edición del cielo e iluminación del entorno
- Integración de sonidos de la naturaleza
- Incorporación de las gafas Oculus Rift y los mandos Oculus Touch

3. Herramientas utilizadas

Para el proyecto final se han utilizado seis herramientas: Blender, Photoshop CS6, NormalMap Online, Substance Painter, World Machine y Unreal Engine 4.

3.1. Blender 2.79

Blender es un software multiplataforma, gratuito y de código abierto, para la creación de gráficos en 3D.

Con Blender se lleva a cabo el modelado de las diferentes mallas y se obtienen la mayoría de los mapas de textura.

Addons

Se han utilizado algunos addons para facilitar la creación de modelos y materiales:

Node: Node Wrangler

Incorpora herramientas para facilitar la edición de nodos. Por ejemplo, con Ctrl + Shift + Click en un nodo, éste se conecta a un nodo *Viewer* y se visualiza el resultado del mismo. Esto puede ser útil con los nodos de ruido, por ejemplo, ya que se puede ver en qué partes de la malla se está aplicando éste.

Mesh: A.N.T. Landscape

Posee múltiples parámetros para la creación de montañas.

Add Curve: Sapling Tree Generator

Herramienta para la creación de árboles. En el proyecto también se utiliza para modelar los tallos de las flores.

Import – Export: Import Images as Planes:e escoge una imagen y se importa un plano con ella como textura.

Import – Export: FBX format: se exporta en formato FBX.

3.2. Adobe Photoshop CS6

Es un editor de gráficos utilizado especialmente para la creación y tratamiento de imágenes.

Se utiliza para obtener los mapas de opacidad, utilizados con las plantas, flores y hojas, y los mapas de desigualdad.

3.3. NormalMap Online

Es una aplicación en línea (http://cpetry.github.io/NormalMapOnline/) en la que se pueden obtener diferentes mapas de textura a partir de imágenes y mapas de altura.

Con esta herramienta se obtienen los mapas normales de las flores, las hojas y las plantas, y los mapas normal y de oclusión ambiental de los troncos de los árboles.

3.4. Substance Painter

Consiste en un software para pintar en 3D, que permite texturizar, renderizar y exportar a otras herramientas, entre ellas Unreal Engine 4.

Se obtienen los mapas de textura de las montañas y las rocas más grandes.

De manera similar a Photoshop, en Substance se utilizan capas para texturizar.

3.5. World Machine

Esta herramienta permite la creación de mapas de altura de terrenos, modelos y texturas a partir de nodos llamados *devices*. Tiene utilidades avanzadas, como la de añadir erosión o crear montañas, con múltiples parámetros.

Se modela aquí el terreno de la escena.

3.6. Unreal Engine 4

Se trata de un motor de juego, con herramientas para el diseño y construcción de videojuegos y simulaciones.

Se importan los objetos y texturas y se lleva a cabo el resto del desarrollo del proyecto con esta herramienta.

4. Metodología

Se ha utilizado la metodología ágil Kanban. El proyecto se descompone en tareas específicas, que serán tratadas como objetivos a corto plazo.

Las tareas se distinguen entre:

Tareas pendientes: todavía no se han comenzado.

Tareas en curso: se han empezado, pero no acabado.

Tareas para revisar: tareas finalizadas pero que muy probablemente deban ser modificadas en un futuro, normalmente debido a avances en otras tareas.

Tareas finalizadas: totalmente acabadas.

Se estima un tiempo necesario para cada tarea, con cierto margen: tiempo dedicado a aprendizaje y tiempo de resolución de problemas.

También se ha llevado la cuenta del tiempo dedicado para cada tarea. Así se evita permanecer demasiado con una tarea en curso: esto haría que no hubiera un avance visible en el proyecto, y que las tareas pendientes no dispongan de tiempo suficiente.

El trabajo con metodología enfoca en las *tareas en curso*. Una vez se finalice una tarea, pasa a *tareas para revisar* o a *tareas finalizadas*, según el caso, y se añade a *tareas en curso* alguna de las *tareas pendientes*.

Las tareas más importantes se priorizan, por lo que se tiene un producto acabado lo más pronto posible: un entorno 3D en Unreal con iluminación, sonido, variedad de objetos y con las gafas integradas. Después se realizan tareas para mejorar el producto: sistemas de partículas, más objetos, mejor trabajo en los materiales y texturas, etc.

5. Cuerpo del trabajo

5.1. Modelado

Salvo el terreno, que ha sido modelado en World Machine, las diferentes mallas se han realizado con Blender.

Para algunas rocas, se han utilizado pinceles gratuitos de suavizado y de texturas rocosas.

5.2. Materiales y texturas

Se utilizan cuatro herramientas de software para obtener las texturas y aplicarlas a las mallas: Blender, Photoshop, Substance Painter y Unreal Engine 4.

5.2.1. Blender

Cycles Render

Se trabaja con *Cycles Render*, el motor de Blender que renderiza mediante trazado de rayos (*Ray-tracing*) con un resultado mucho más realista que el motor antiguo, *Blender Render*.

Smooth Shading

En todos los casos se aplica como sombreado el *Smooth*. Éste utiliza interpolación lineal entre los vértices, de modo que cada píxel tiene un color propio y no se aprecian bordes.

UV Map

Para cada elemento se crea un mapa UV. En general, en *Edit Mode* se hace mediante U – *Smart UV Project*. Para las rocas y las montañas, al tratarse de mallas muy grandes, para tener más control sobre el mapa, se seleccionan ciertos segmentos de éstas y se hace *Mark Seam* en el panel de la izquierda, ventana *Shading/UVs* o bien con Ctrl + E. En estos casos se utiliza U – *Unwrap*.

Texturizado

El texturizado se lleva a cabo mediante materiales de nodos, realizados en la ventana *Node Editor*.

Los shaders *BSDF* (*Bidirectional Scattering Distribution Function*) describen la luz reflejada, refractada y absorbida en la superficie de un objeto.

Los principales nodos utilizados son:

- *Add Shader*: para cada píxel se suman los valores de los dos *shaders* mezclados.
- *Bump*: genera una normal a partir de una textura de altura, haciendo que la superficie no se aprecie completamente lisa.
- *Color Ramp*: se mapean valores a colores por medio de un gradiente.
- *Diffuse BSDF*: la reflexión difusa de la luz.
- *Glossy BSDF*: cantidad de brillo.
- *Image Texture*: la imagen importada desde el explorador de archivos. También se utiliza este nodo para las imágenes obtenidas con *Bake*.
- *Mapping*: editar valores del mapeado, como por ejemplo la escala en cada eje.
- *Material Output*: coloca el resultado de la información del material sobre la superficie del objeto.
- *Color Mix RGB*: se combinan dos colores según el peso determinado por el factor.
- *Mix Shader*: mezcla dos *shaders* según el peso determinado por el factor.

- *Noise Texture*: añade ruido.
- *RGB Curves*: correcciones de color para cada canal.
- *Subsurface Scattering*: hace que algo de luz entre en el objeto y luego salga dispersa, en lugar de reflejarla toda.
- *Texture Coordinate*: coordenadas de textura. Se utiliza *UV* como input del vector del nodo *Mapping*.
- *Translucent BSDF:* transmisión difusa de la luz.
- *Transparent BSDF*: añade transparencia a partir del canal alfa como factor.

Con el *Viewport* en *Render* se visualiza cómo queda el material sobre el objeto.

Una vez finalizados los materiales, se utiliza el método *Bake* para obtener los diferentes mapas de textura. Éste se encuentra en *Propierties – Render* del objeto seleccionado.

Los tipos de *Bake* utilizados son los siguientes:

- *Ambient Occlusion*: se obtiene el mapa *ambient occlusion*.
- *Diffuse:* se obtienen los mapas de color base de las texturas.
- *Glossy*: los mapas obtenidos son la inversa (en blancos y negros) de los mapas *roughness*, por lo que se invierten en Photoshop.
- *Normal*: se utiliza para comprobar que una malla *Low Poly* parece la *High Poly* al tener en su material el mapa normal, y para obtener los mapas normales de las rocas más pequeñas (las texturizadas en Blender y no en Substance Painter).
- *Subsurface*: se obtienen los mapas *Translucency* de las hojas, flores y plantas.

5.2.2. Adobe Photoshop CS6

Se obtienen los mapas de opacidad y de desigualdad.

El mapa de opacidad se obtiene a partir de la textura de color base: lo que se quiere visualizar se pinta en blanco, y lo que se desea transparente, en negro (fondo).

Para el mapa de desigualdad, se parte del mapa *glossy* y se invierten el blanco y el negro.

5.2.3. NormalMap Online

Se obtienen los mapas normales de hojas, plantas y flores, y los mapas *normal* y *ambient occlusion* de los troncos de los árboles.

Se arrastra la imagen de color base y se escoge el mapa deseado.

5.2.4. Substance Painter

Se utiliza con las montañas y algunas de las rocas, cada una de las cuales tiene una malla *High Poly* y una *Low Poly*.

Al comenzar un nuevo proyecto se importa la malla *Low Poly*. Aquí se puede escoger entre diferentes plantillas y resoluciones para las imágenes. Se selecciona la plantilla de Unreal Engine 4 y resolución de 4096.

Después *en Texture Settings – Bake Mesh Maps – Normal* se añade la malla *High Poly* y se hace el *Bake* de las texturas. La malla *Low Poly* ahora con el mapa normal generado se ve como la *High Poly*.

5.2.5. Unreal Engine 4

Cada malla tiene un cierto número de *Material Slots*, según la cantidad de materiales que tuviera el objeto en Blender. Aquí se colocan las instancias de material correspondientes. Para ver a qué parte del material pertenece cada *slot*, basta con activar *Resaltar*, que marca en amarillo esta parte, o *Isolate*, que deja de mostrar las demás.

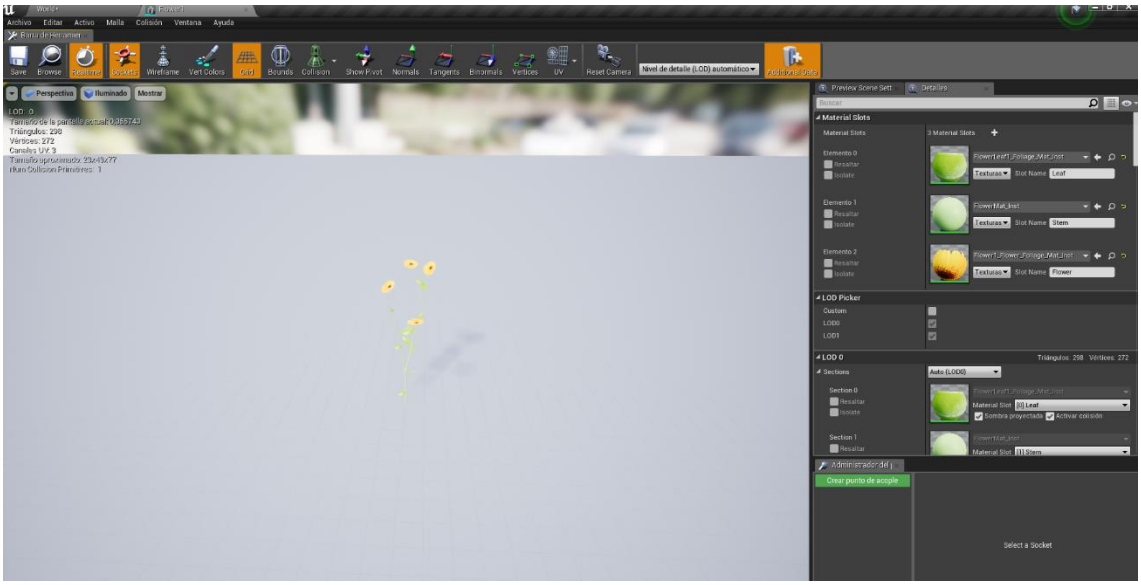

Figura 12: Ventana de las mallas estáticas

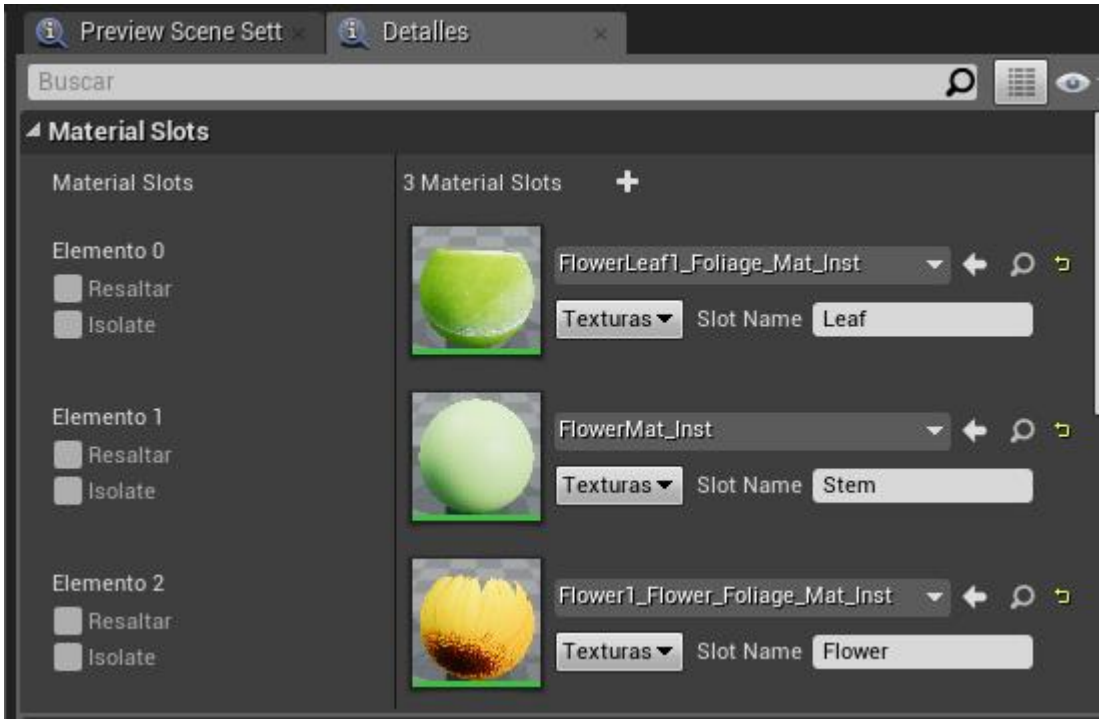

Figura 13: Panel de *Material Slots* en la ventana de las mallas estáticas

Al igual que en Blender, los materiales se crean a partir de nodos. Se utilizan los siguientes tipos de nodo:

- *Add*: suma los valores de entrada.
- *Append*: se combinan canales en un sólo vector.
- *Clamp*: hace los valores de entrada tengan un valor máximo y uno mínimo.
- *Constant*: contiene un valor en coma flotante.
- *Landscape Layer Blend*: mezcla diferentes texturas o materiales para ser utilizados como capas en el terreno.
- *Linear Interpolate (Lerp)*: mezcla dos valores de entrada de acuerdo a un tercer valor, utilizado como máscara.
- *Multiply*: se multiplican los valores de entrada.

- *Power*: se multiplica la entrada *base* por sí misma cierto número de veces, de acuerdo a la otra entrada (*exponent*).
- *Result node*: el nodo con el resultado de material, que aparece cada vez que se crea un material nuevo. Tiene los inputs de *color base*, *especular*, *desigualdad*, etc.
- *Scalar Parameter*: parámetros con un valor escalar que se puede modificar en las instancias.
- *Simple Grass Wind*: añade moviento a las mallas, simulando viento. Se utiliza con las plantas, las flores y las hojas de los árboles.
- *Texture Coordinate*: tiene como salida coordenadas UV en un vector de dos canales. Con este nodo se puede modificar el tamaño de las coordenadas de textura.
- *Texture Sample*: contienen la imagen de la textura. Se convierten a parámetro para poder cambiar esta imagen en las instancias.
- *Vector Parameter*: un vector con valores en cada canal. Se usa para añadir un color en las instancias.

Cabe destacar dos detalles importantes en cuanto a los nodos *Texture Sample* (Figura 54):

- En *Tipo de muestreador* se debe seleccionar el correspondiente a la textura: *color* en caso de las de color (*albedo*), *normal* para las normales, *máscaras* para las de *roughness* y *color lineal* para la *AO*/*Rough*/*Metallic* obtenida desde Substance. En caso contrario habrá un error de compilación.
- En las texturas del material del terreno, en *Sampler Source* se escoge *Shared: Wrap*. Para el resto de las texturas, se deja el valor por defecto: *From Texture Asset.*

Muchos de los objetos comparten características similares en cuanto al material se refiere. Las diferencias entre ellos están únicamente en las texturas y en algunos parámetros tales como la cantidad de brillo *specular* o de desigualdad (*roughness*).

Hacer cambios en los materiales requiere que éstos tengan que recompilarse, algo poco eficiente si lo hiciéramos para cada uno de los objetos de la escena. Esto se soluciona utilizando instancias: permiten, utilizando un mismo material, cambiar diferentes parámetros para cada objeto.

Se crean ocho materiales en el proyecto, denominados *Master Materials*:

Opaque_Master_Mat

Es el material utilizado para las rocas de tamaño mediano, como las que tienen musgo.

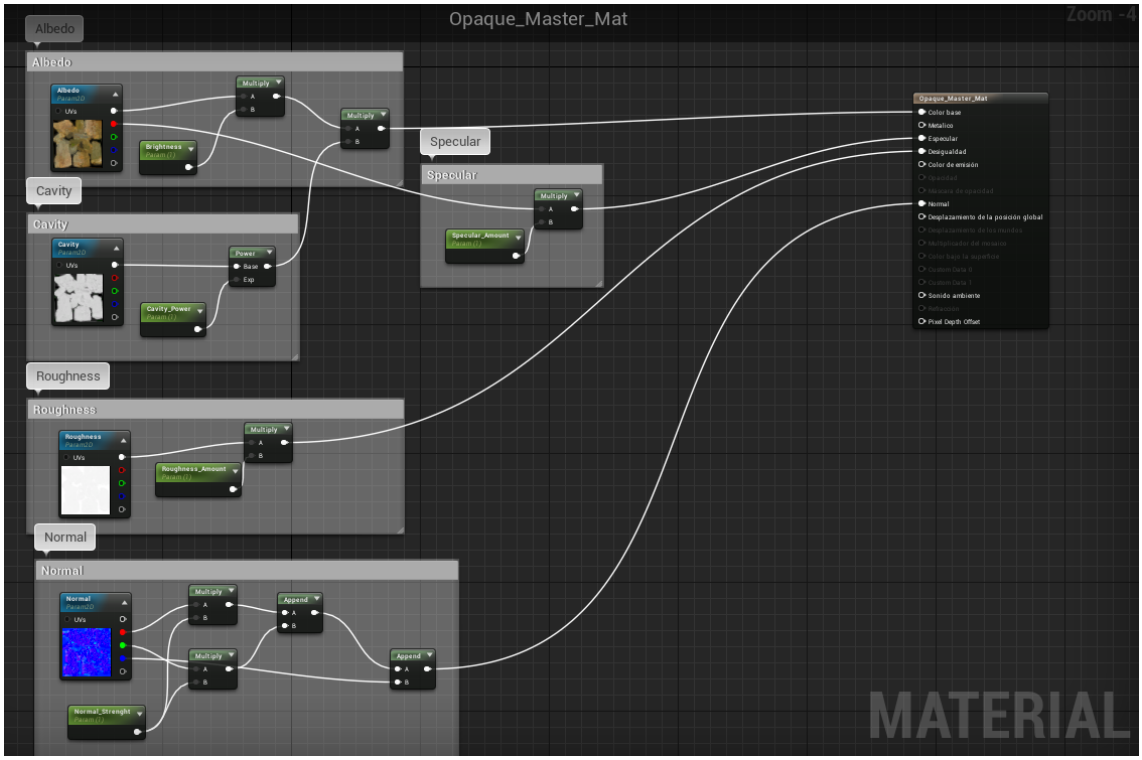

Figura 14: *Opaque_Master_Mat*

Consta de cinco bloques diferenciados: *albedo*, *cavity*, *roughness*, *normal* y *specular*.

En *albedo* se encuentra la textura de color difusa (en Blender se obtiene con *Bake - Diffuse*). Se le añade un brillo y el resultado se multiplica con el mapa de *Ambient Occlusion* (bloque *cavity*) y se lleva al input *base color*. El *specular* se obtiene a partir del canal rojo de la textura de *albedo*: el brillo no será uniforme en todo el objeto.

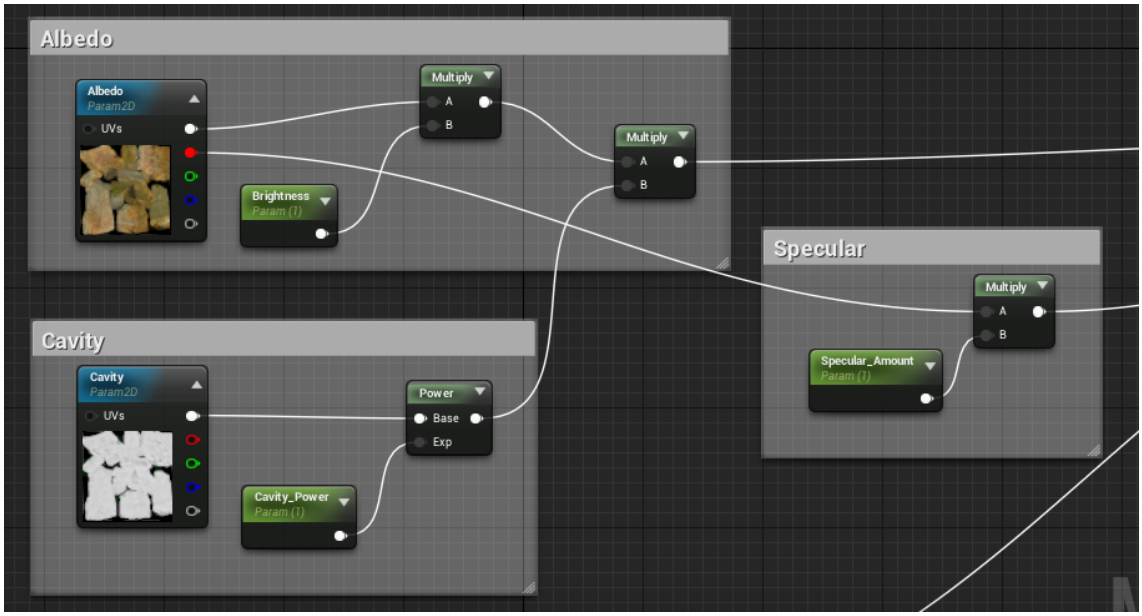

Figura 15: Parte de *Opaque_Master_Mat* con los bloques *albedo*, *cavity* y *specular*

La *desigualdad* se obtiene a partir del mapa *roughness* (en Blender se parte de *Bake – Glossy* y en Photoshop se invierten el blanco y el negro).

Para la *normal*, los canales rojo y verde se multiplican por un parámetro. Posteriormente mediante nodos *append* se vuelven a unificar los tres canales.

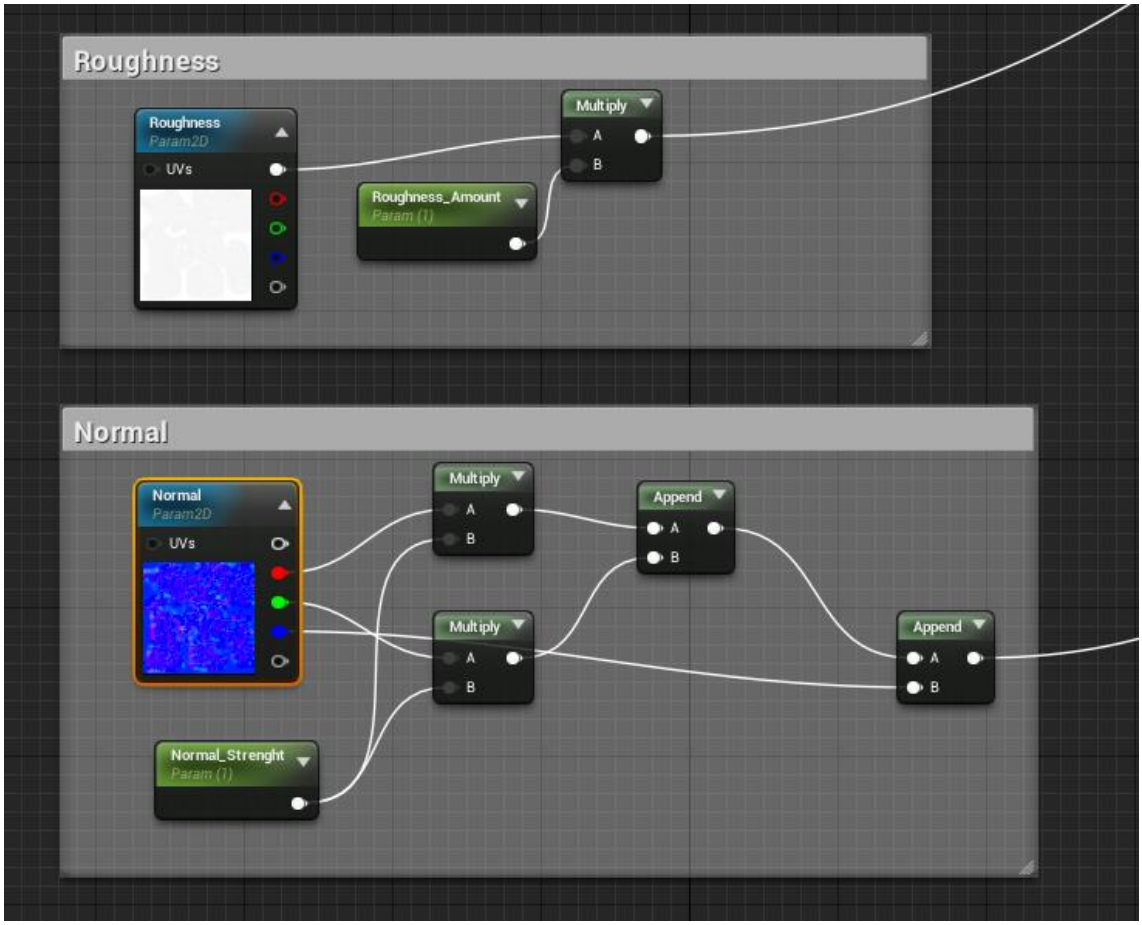

Figura 16: Parte de *Opaque_Master_Mat* con los bloques *roughness* y *normal*

Las instancias creadas a partir de este material se ven de este modo:

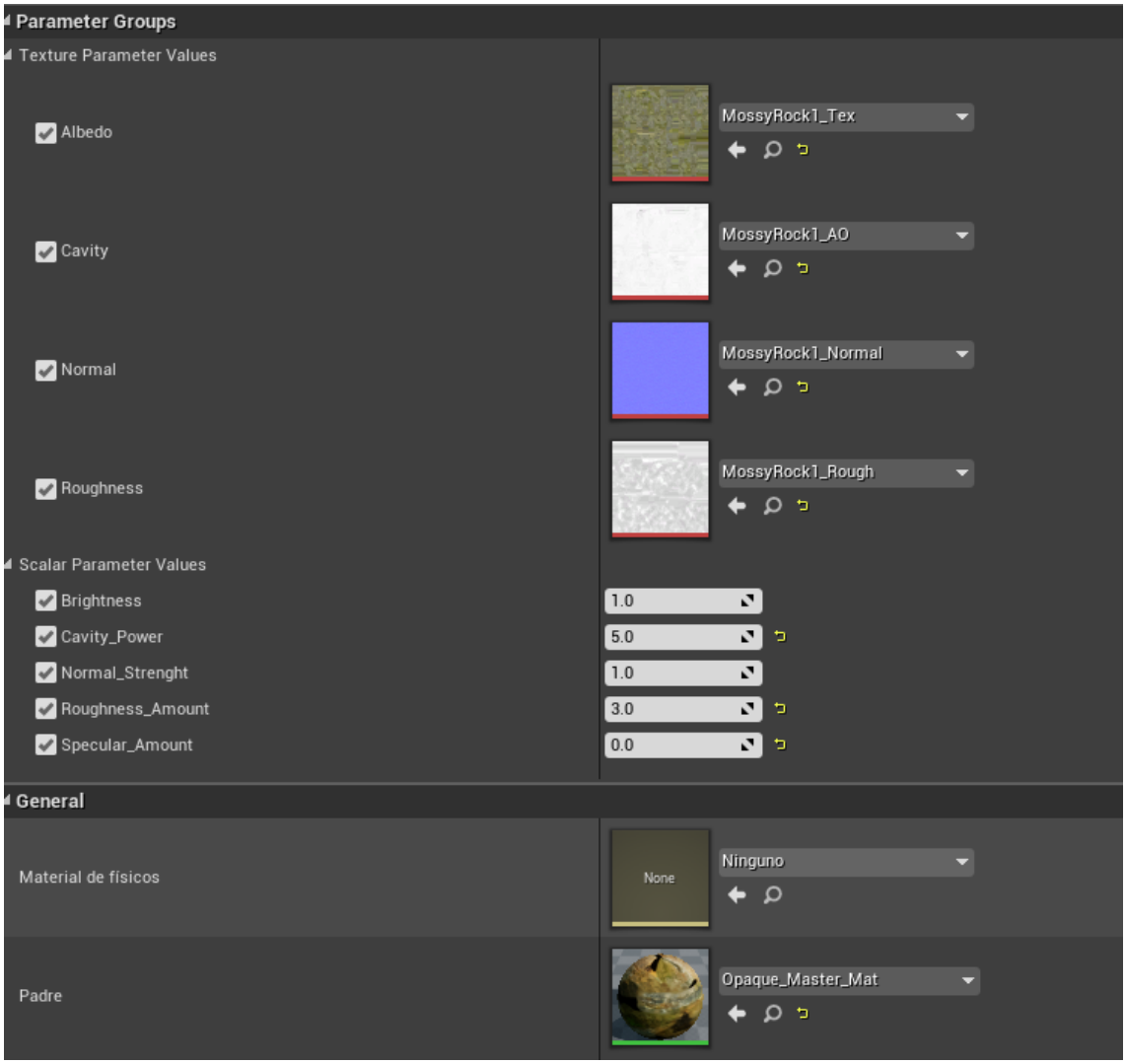

Figura 17: Instancia de *Opaque_Master_Mat*

Cada instancia de este tipo tiene sus propios mapas de textura, así como valores de cantidad de brillo, desigualdad, etc.

SmallOpaque_Master_Mat

Se utiliza una versión más simple de *Opaque_Master_Mat* para las piedras y ramitas, los elementos más pequeños del entorno. Utiliza únicamente el *color base* y la *normal*.

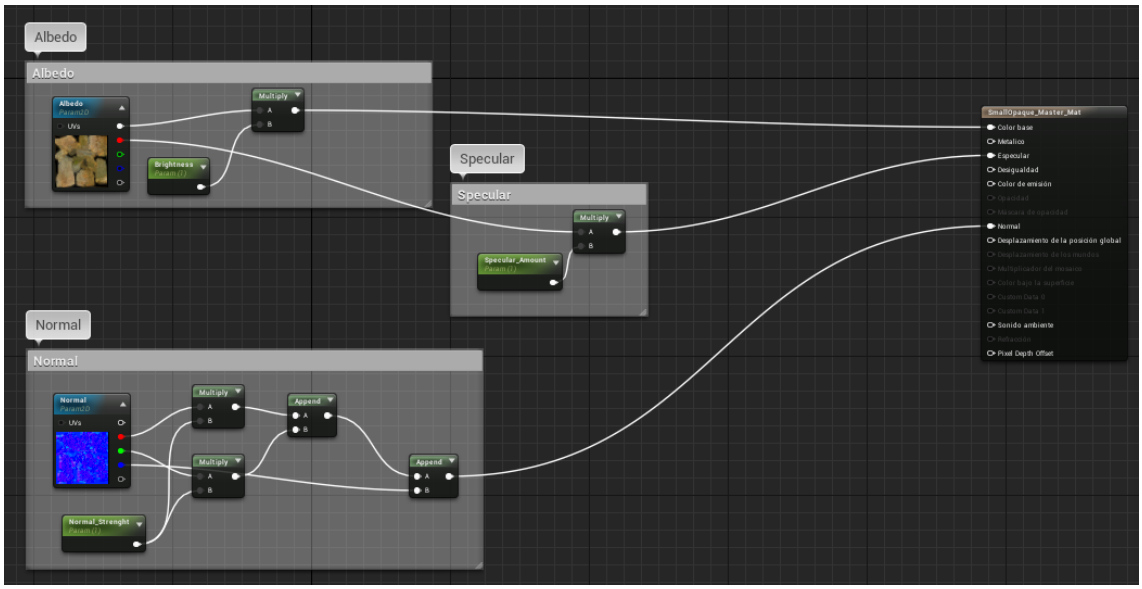

Figura 18: *SmallOpaque_Master_Mat*

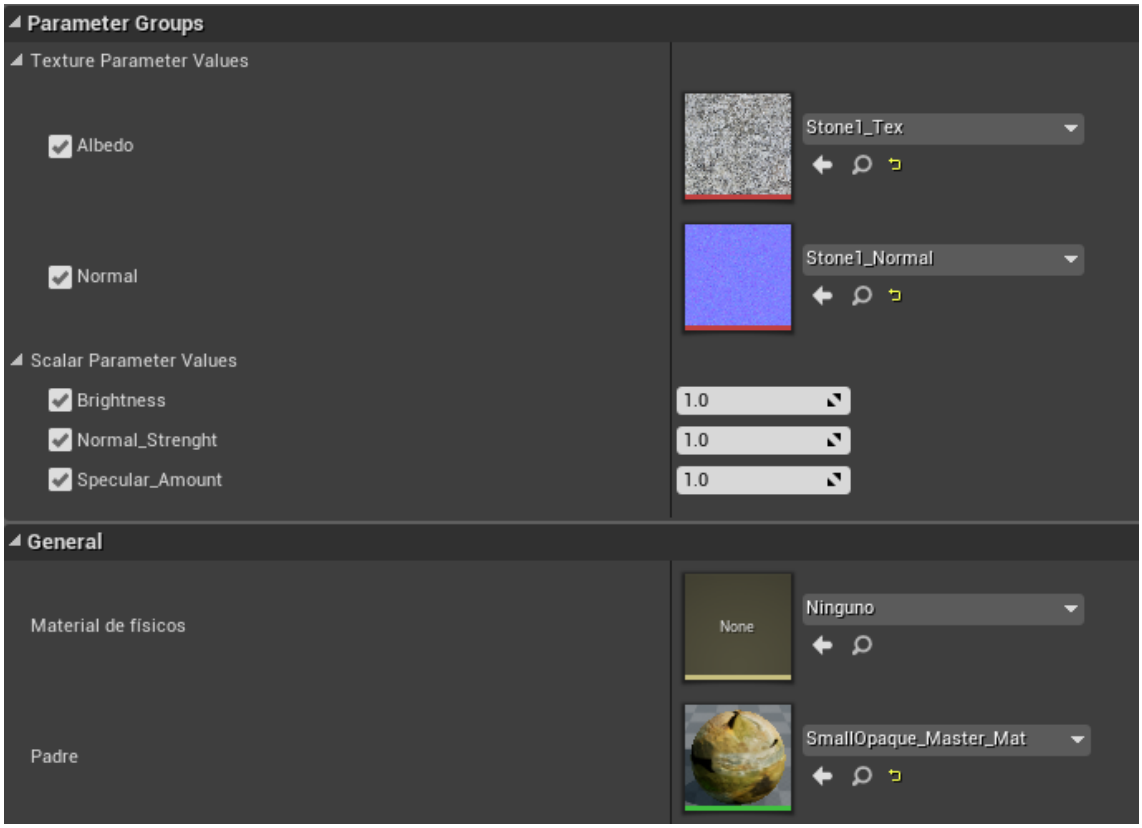

Figura 19: Instancia de *SmallOpaque_Master_Mat*

OpaqueSubs_Master_Mat

Este material es el utilizado por las rocas más grandes y las montañas, es decir, los elementos texturizados en Substance Painter.

Las tres imágenes obtenidas en Substance Painter son las de *base color*, *normal* y una tercera con tres mapas, uno en cada canal RGB: *ambient occlusion*, *roughness* y *metallic*. Se unen a los inputs correspondientes y además se añade una constante con valor 0 para el brillo especular.

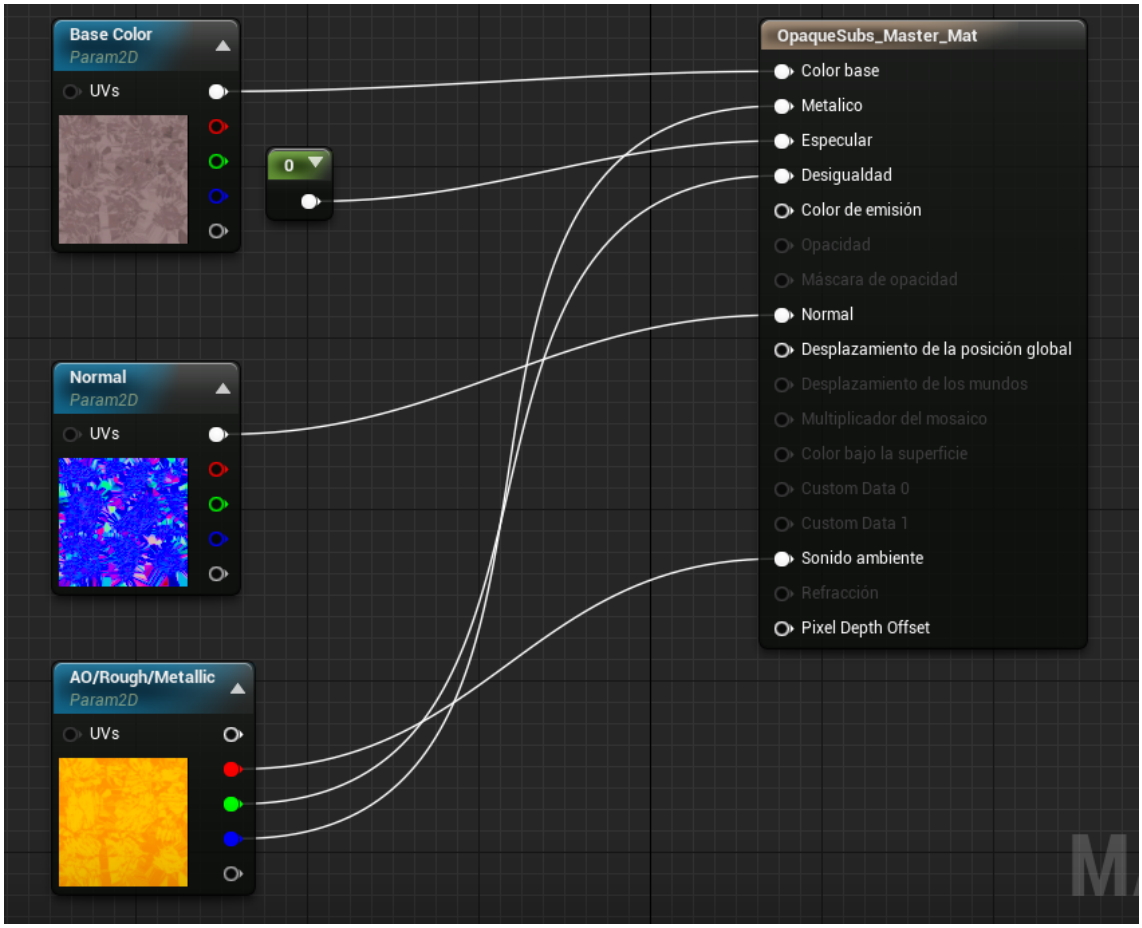

Figura 20: *OpaqueSubs_Master_Mat*

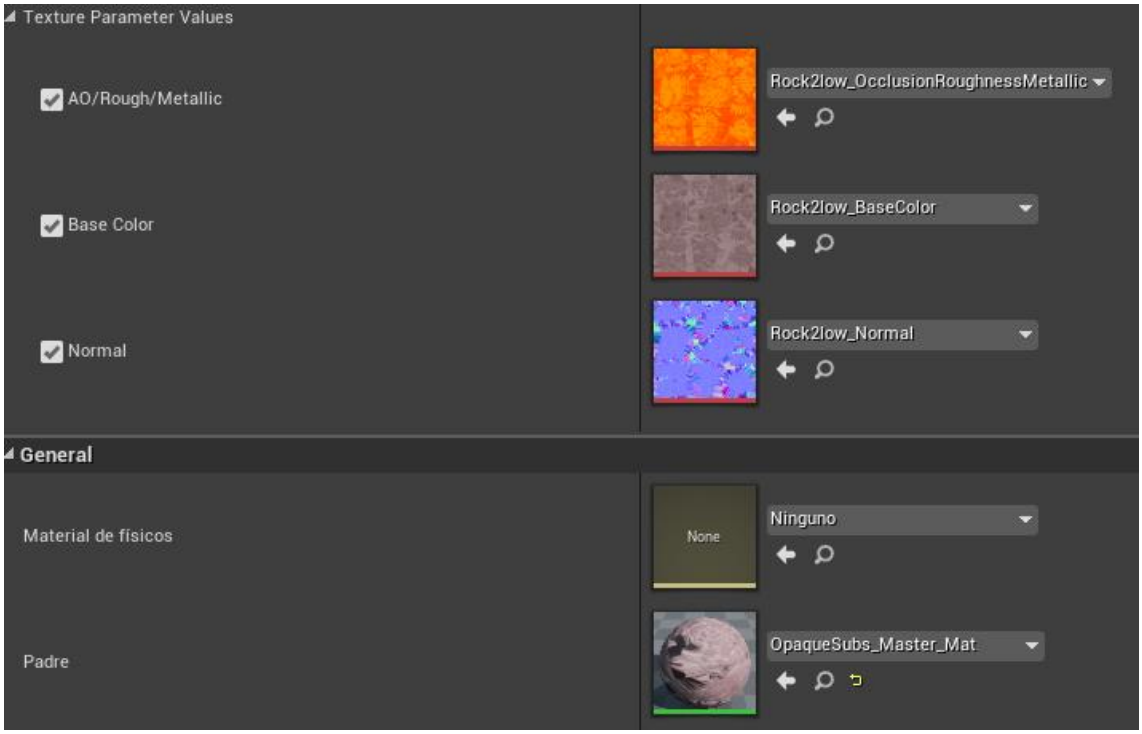

Figura 21: Instancia de *OpaqueSubs_Master_Mat*

Foliage_Master_Mat

Se trata del material de las hojas, las plantas y las flores: objetos planos. Es similiar al *Opaque_Master_Mat*, pero con algunos cambios:

- En las propiedades del nodo resultado del material se debe seleccionar:

  En *Modo de la combinación* la opción *Enmascarado*, para que las texturas tengan una parte transparente (el fondo).

  En *Modelo de degradación* la opción *Two Sided Foliage*. Esto hará que las texturas se vean por las dos caras del plano. También se activa *Dos caras*.

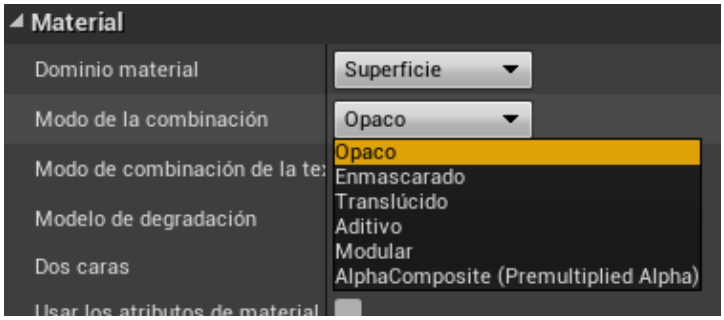

Figura 22: Detalle del *Modo de la combinación* en las propiedades de los materiales

Figura 23: Detalle del *Modelo de degradación* en las propiedades de los materiales

- No hay *cavity*, por lo que la textura en *albedo* con su correspondiende brillo se enlaza directamente a *color base*.
- Bloque *opacity*: utiliza el mapa de opacidad creado mediante Photoshop y se lleva al input *Máscara de opacidad*. Tiene un parámetro que determina la cantidad de transparencia, siendo 1 su valor por defecto.
- Bloque *subsurface scattering*: aquí se pone el mapa obtenido en Blender con *Bake – Subsurface*, que va dirigido al input *Color bajo la superficie*.

- Bloque *wind*: se introduce una función que simula viento, y dos parámetros para regular el movimiento sobre la superficie de los objetos. Se dirige al input *Desplazamiento de la posición global*.

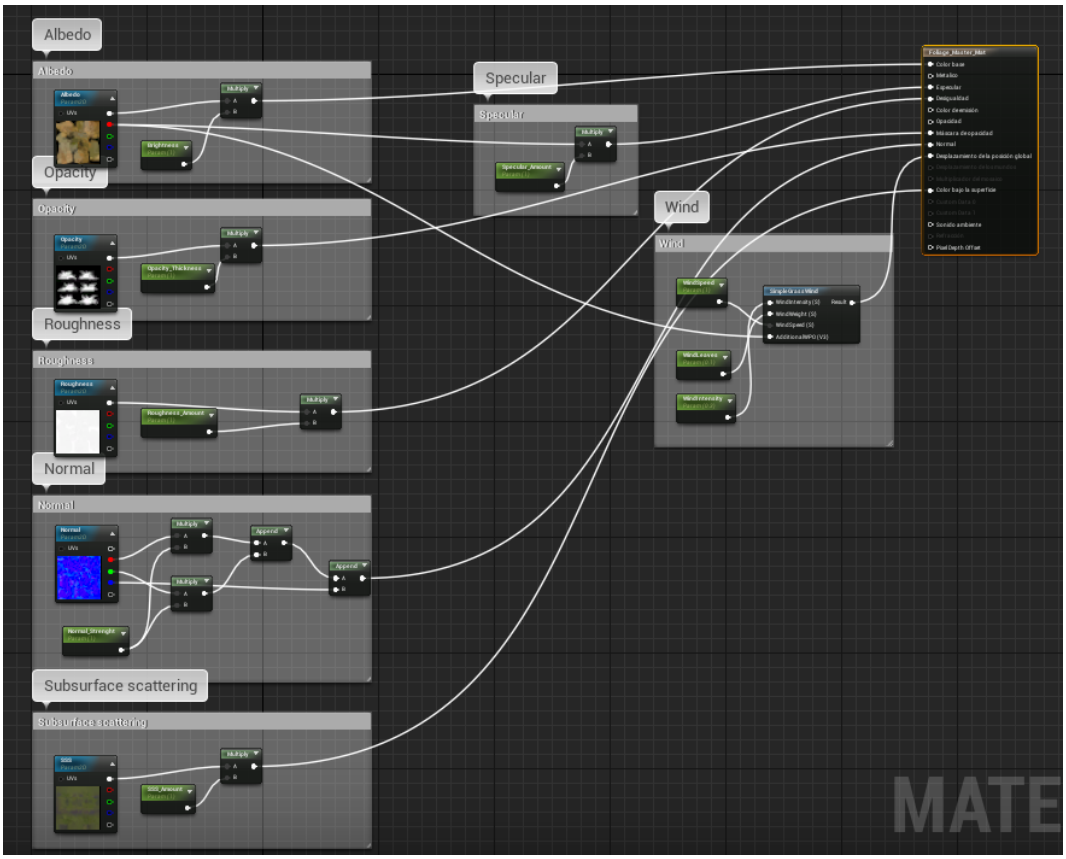

Figura 24: Foliage_Master_Mat

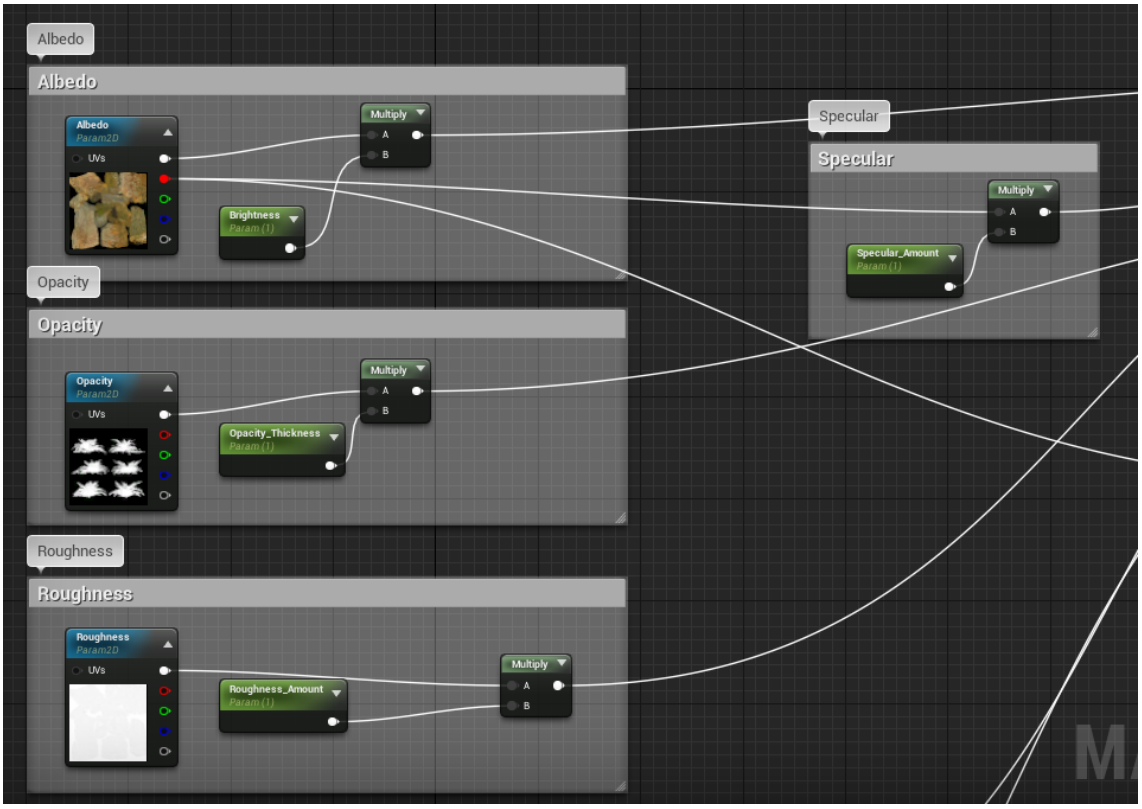

Figura 25: Parte de *Foliage_Master_Mat* con los bloques *albedo*, *opacity*, *roughness* y *specular*

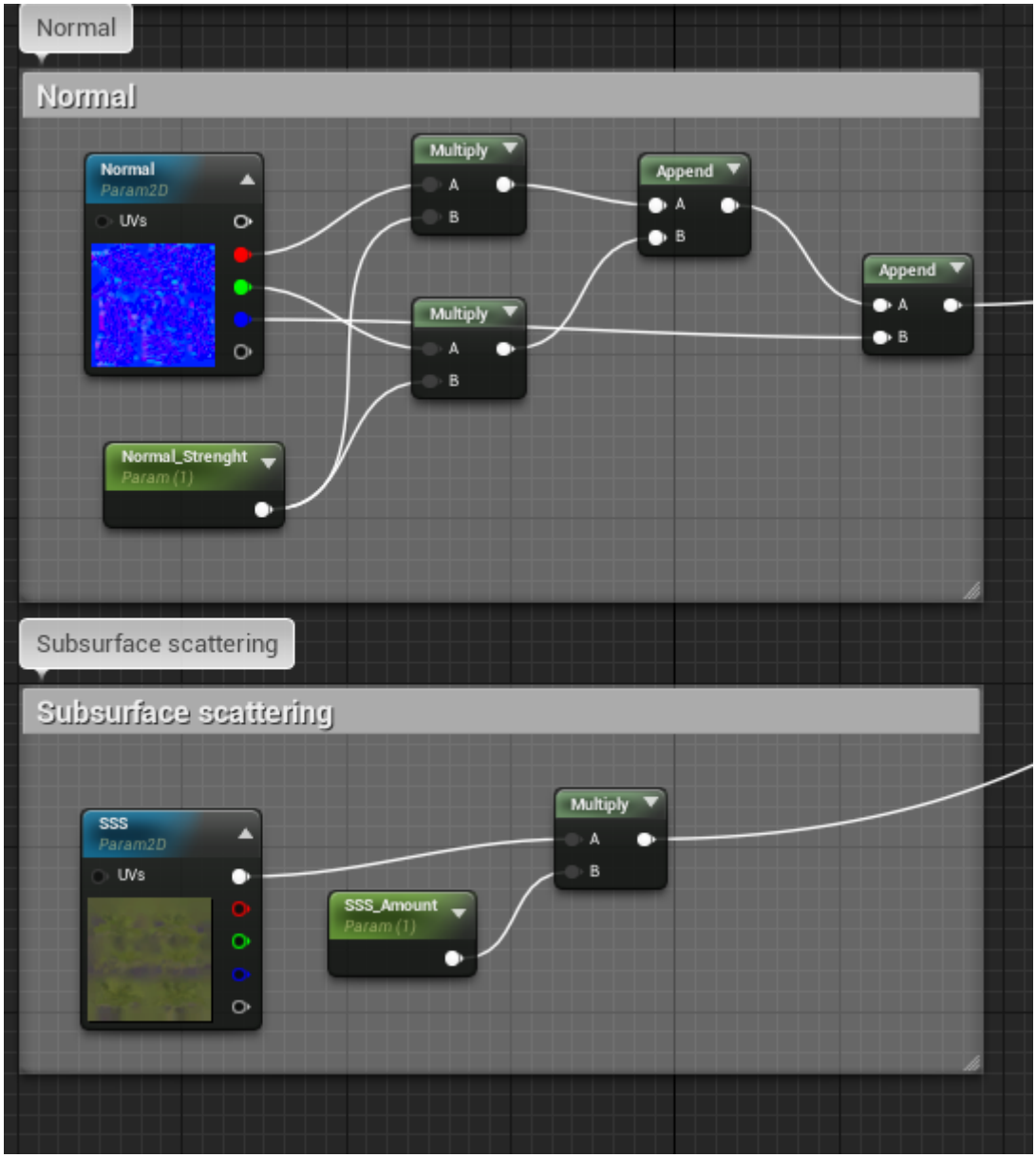

Figura 26: Parte de *Foliage_Master_Mat* con los bloques *normal* y *subsurface scattering*

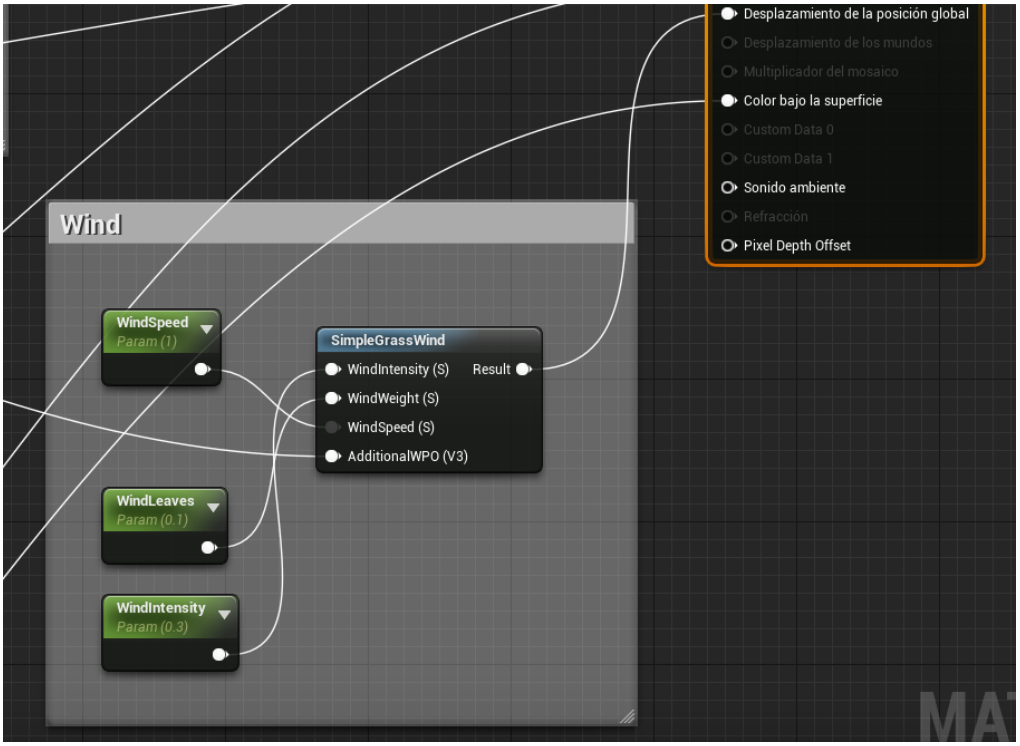

Figura 27: Parte de *Foliage_Master_Mat* con el bloque *wind*

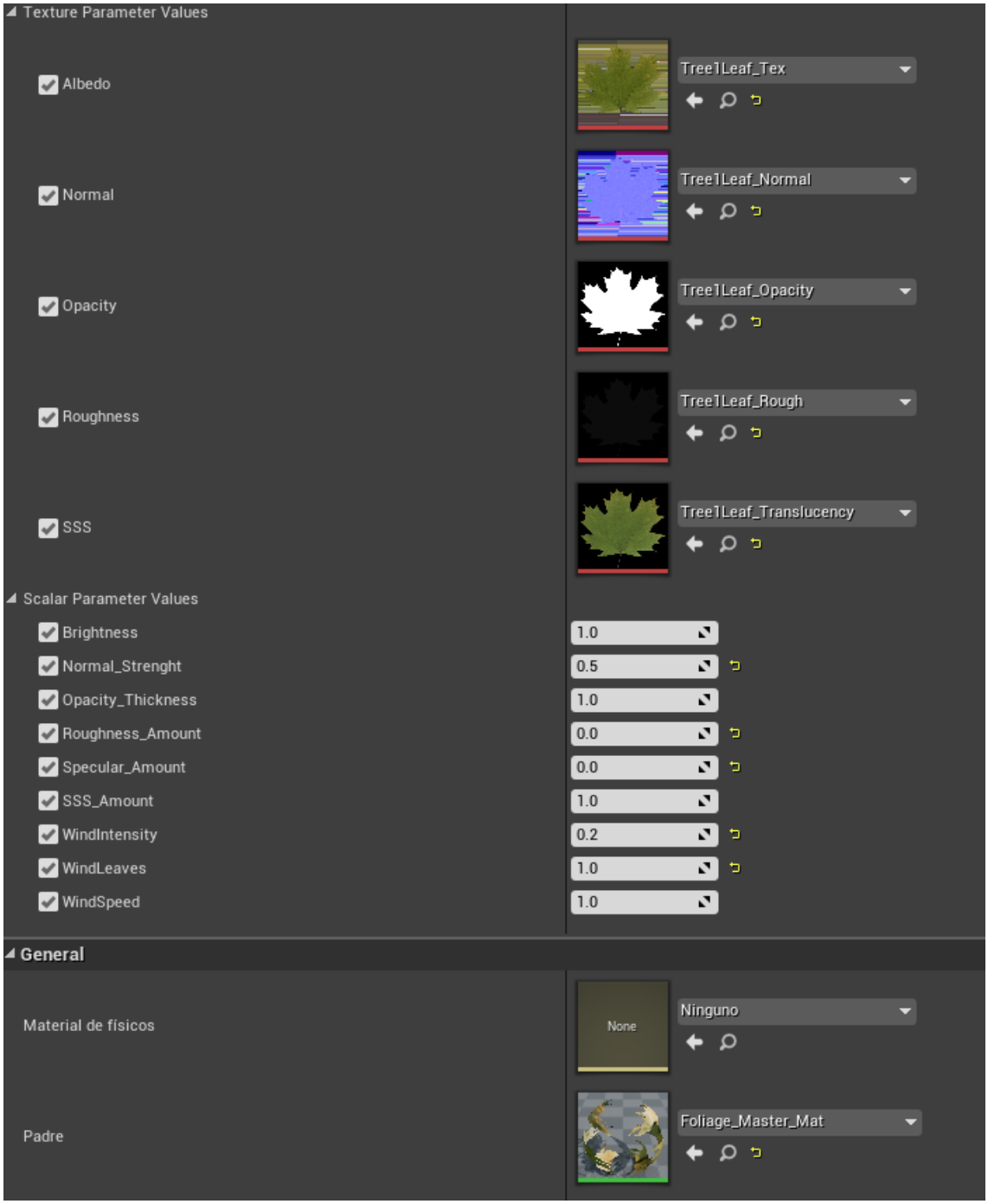

Figura 28: Instancia de *Foliage_Master_Mat*

Como se ve en la figura 28, en estas instancias también se cambian los parámetros de viento, de modo que se puede hacer que éste afecte más a algunas plantas, flores u hojas que a otras.

FarFoliage_Master_Mat

Es el material utilizado por los *LODs* más simples de los árboles y arbustos, que consisten en un par de planos con una imagen.

Tienen únicamente un bloque de *albedo* con la imagen y un brillo, unido a *color base*. El material es, al igual que el anterior, *Enmascarado* y *Two Sided Foliage*. La *máscara de opacidad* viene dada por el canal alfa de la textura.

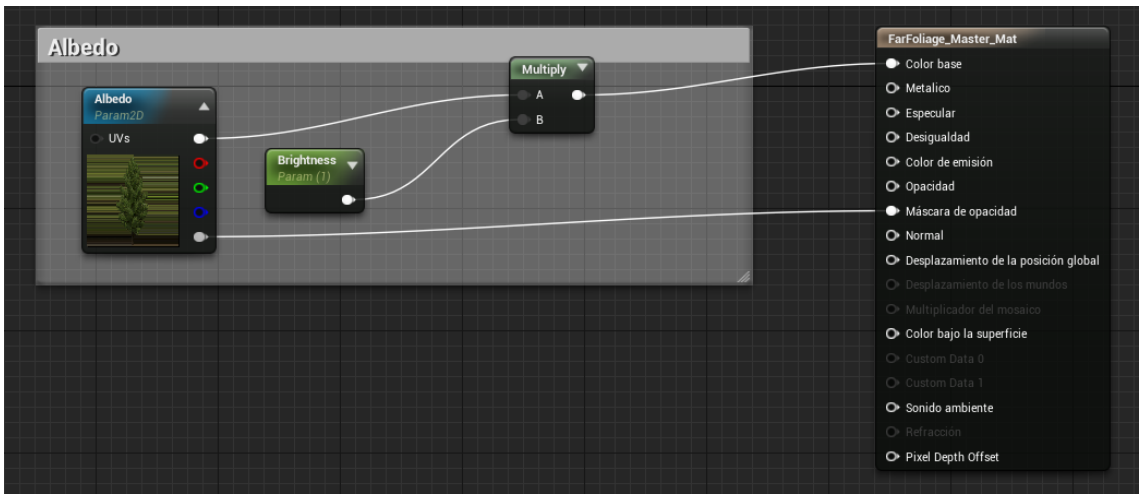

Figura 29: FarFoliage_Master_Mat

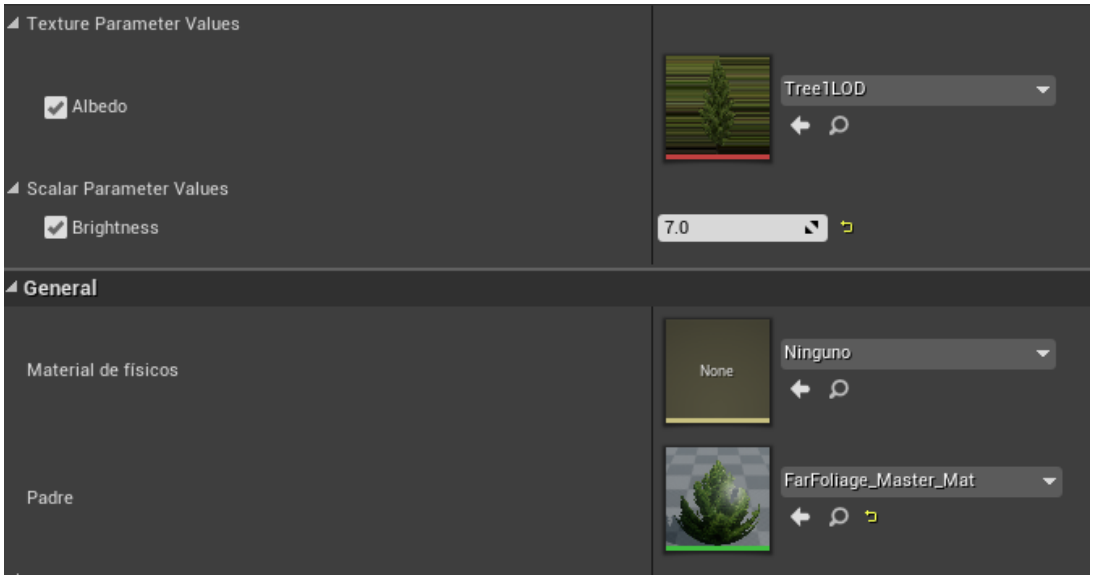

Figura 30: Instancia de *FarFoliage_Master_Mat*

Terrain_Master_Mat

Se trata del material creado para el terreno (5.4.1. Terreno). Se crea una instancia con los diferentes colores a añadir para cada una de las capas de pintura:

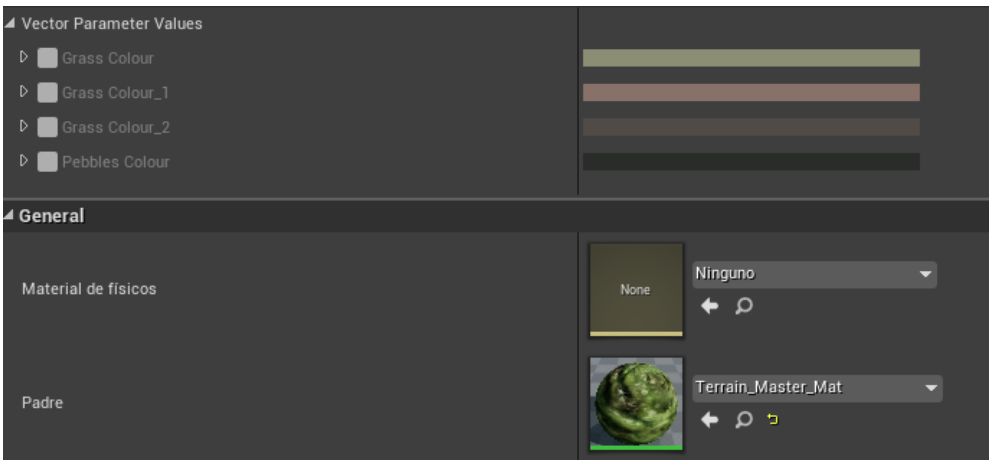

Figura 31: Instancia de *Terrain_Master_Mat*

FallingLeaf_Master_Mat

El material de la hoja que cae mediante un sistema de partículas. Se le puede variar el color.

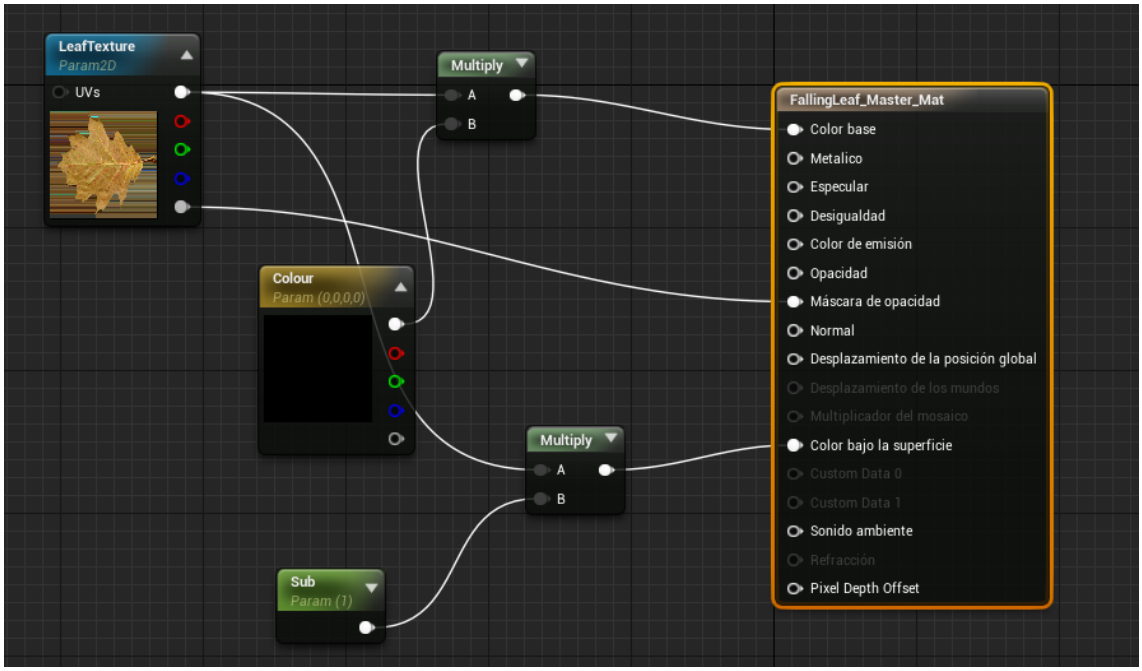

Figura 32: *FallingLeaf_Master_Mat*

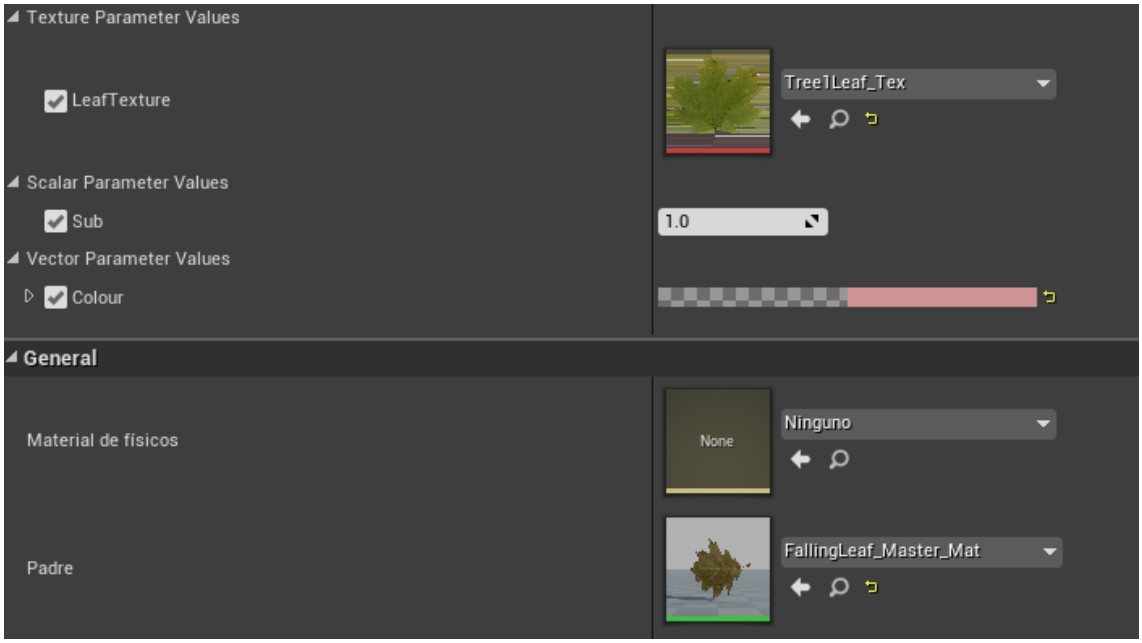

Figura 33: Instancia de *FallingLeaf_Master_Mat*

TreeBark_Master_Mat

Este material se utiliza con los troncos y ramas de los árboles. Tiene texturas de *albedo*, *ambient occlusion* y *normal*.

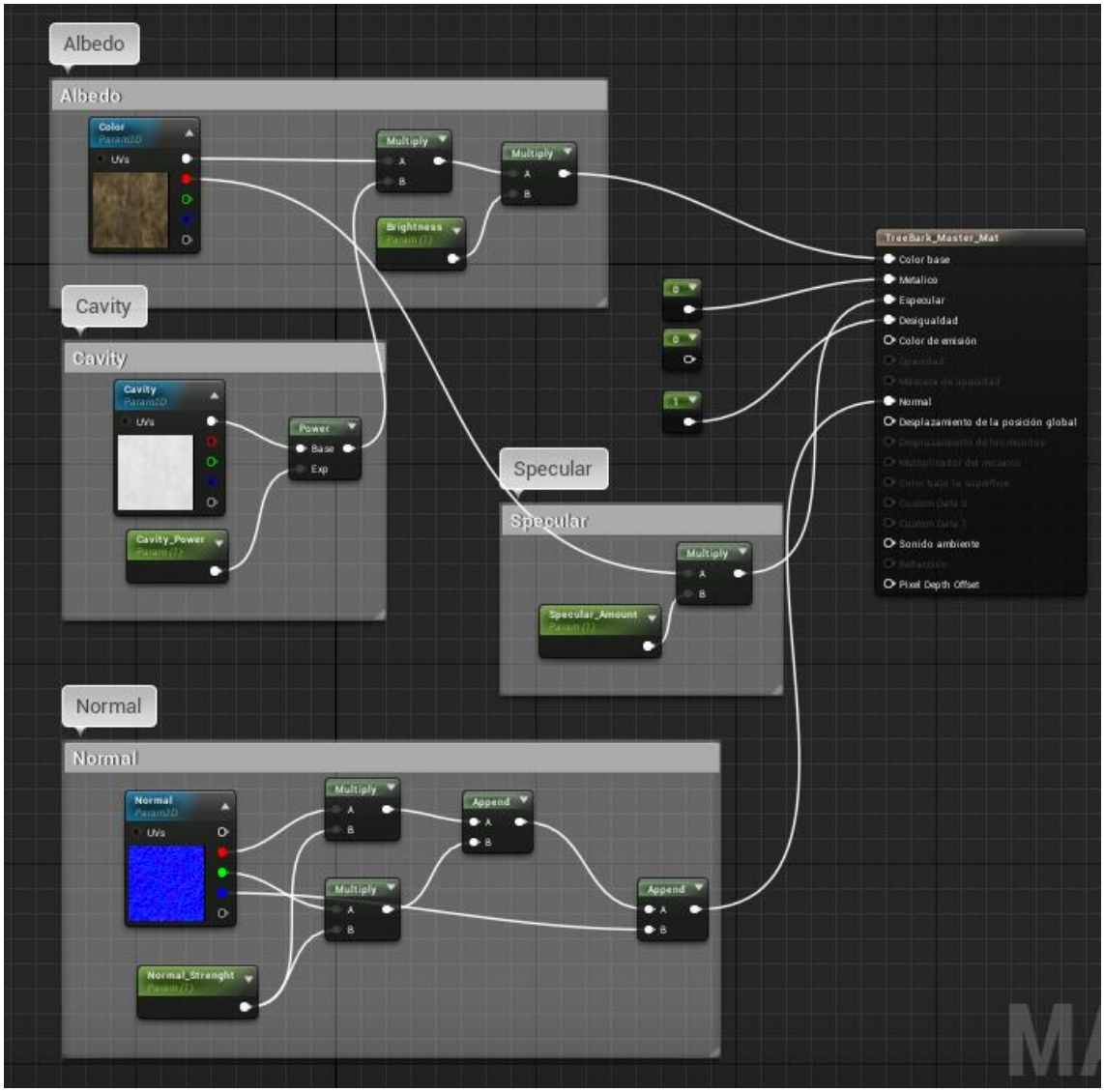

Figura 34: *TreeBark_Master_Mat*

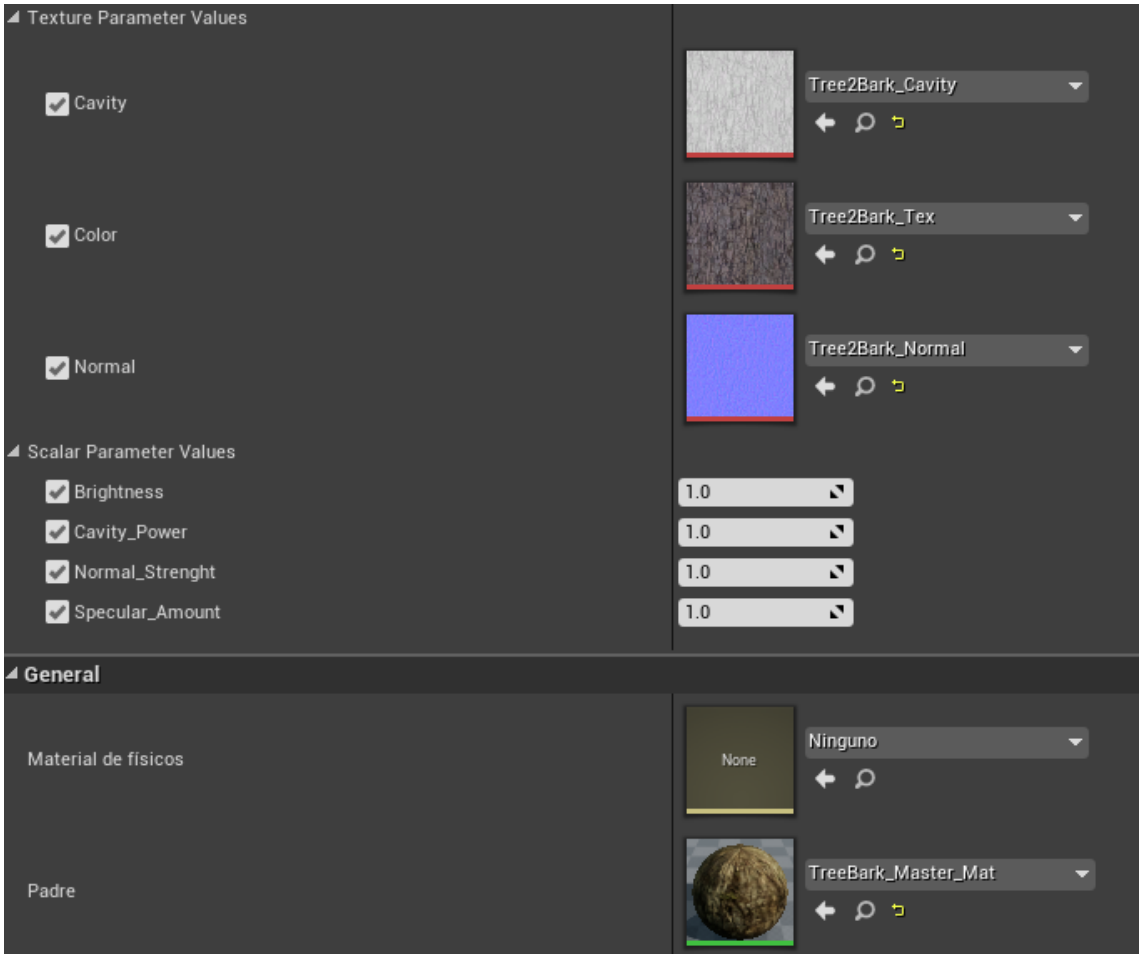

Figura 35: Instancia de *TreeBark_Master_Mat*

5.3. Exportación e importación: de Blender a UE4

Blender

Se utiliza el formato FBX para exportar las mallas a Unreal.

Se selecciona la malla o las mallas a exportar, y en las opciones se activa *Selected Objects* para exportar únicamente esas y no todo lo que hay en el archivo *.blend*.

Como tipo de *Smoothing* se elige *Edge*.

UE4

Al importar aparece un menú de opciones de importación FBX. Se pueden importar los materiales y texturas de las mallas, aunque no es necesario ya que se vuelve a crear el material en Unreal, ahora a partir de los diferentes mapas de textura.

Sólo en el caso del pájaro, se escoge en *Vertex Colors Import Option* la opción *Replace* ya que este objeto tiene *vertex colors*. Para el resto se deja en la opción por defecto, *Omitir*.

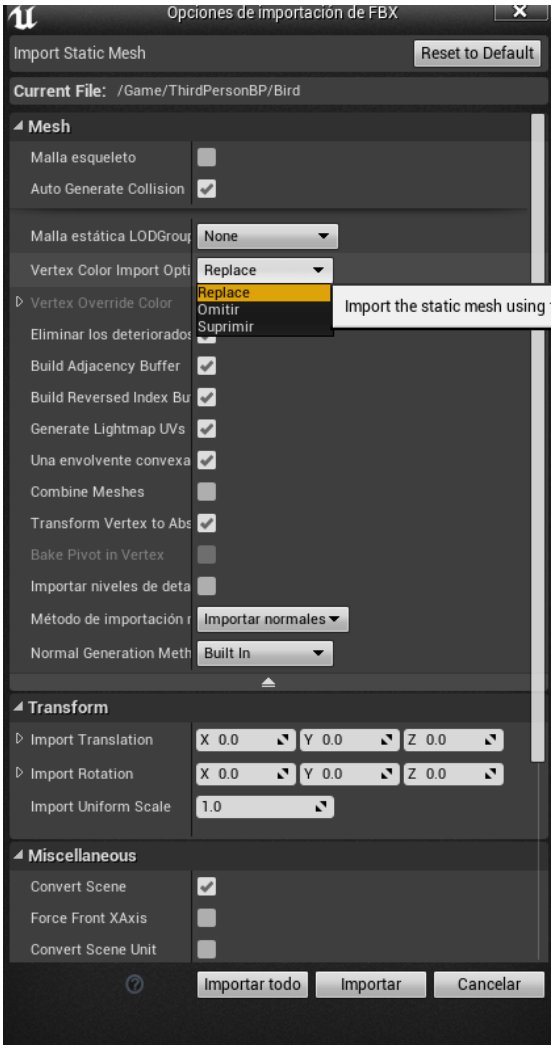

Figura 36: Opciones de importación FBX en UE4

5.4. Elementos creados

5.4.1. Terreno

Modelado

En un principio se realizó en Unreal mediante el modo de *Manipular* en *Terreno* y seleccionando *Crear nuevo*. Para escoger el valor de los parámetros tuve en cuenta las dimensiones recomendadas por la documentación del motor:

Recommended Landscape Sizes

In order to make things easier, here are a number of sizes that maximize the area while minimizing the number of Landscape components.

Overall size (vertices)	Quads / section	Sections / component	Component size	Total Components
8129x8129	127	4 (2x2)	254x254	1024 (32x32)
4033x4033	63	4 (2x2)	126x126	1024 (32x32)
2017x2017	63	4 (2x2)	126x126	256 (16x16)
1009x1009	63	4 (2x2)	126x126	64 (8x8)
1009x1009	63	1	63x63	256 (16x16)
505x505	63	4 (2x2)	126x126	16 (4x4)
505x505	63	1	63x63	64 (8x8)
253x253	63	4 (2x2)	126x126	4 (2x2)
253x253	63	1	63x63	16 (4x4)
127x127	63	4 (2x2)	126x126	1
127x127	63	1	63x63	4 (2x2)

Figura 37: Tamaños recomendados para el terreno

Fuente: https://docs.unrealengine.com/en-us/Engine/Landscape/TechnicalGuide

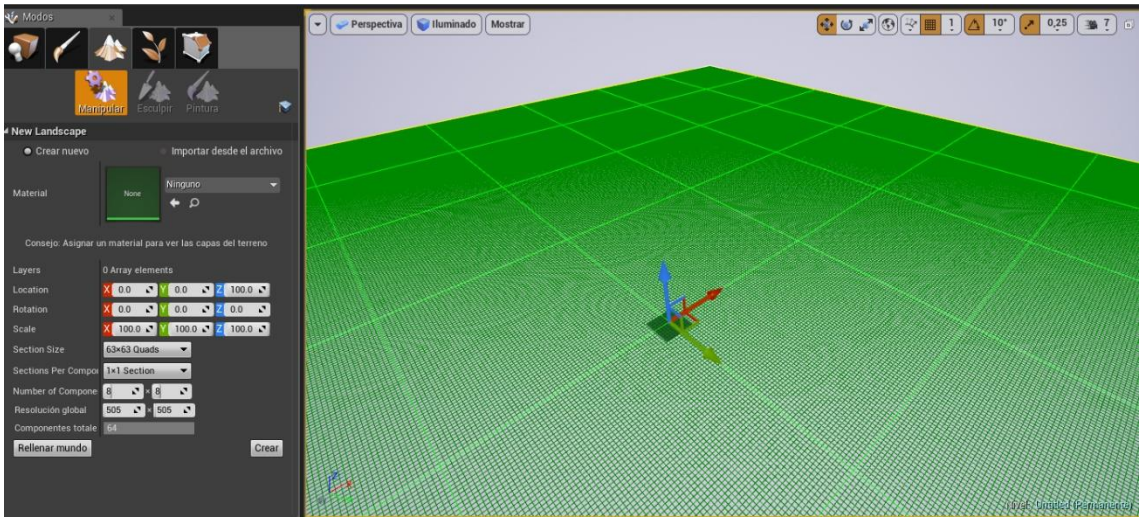

Figura 38: Creación de terreno en UE4

Después, el terreno puede moldearse en el modo *Esculpir* dentro de *Terreno*. Éste posee varias herramientas, como *Esculpir*, *Suavizado*, *Allanar*, etc.

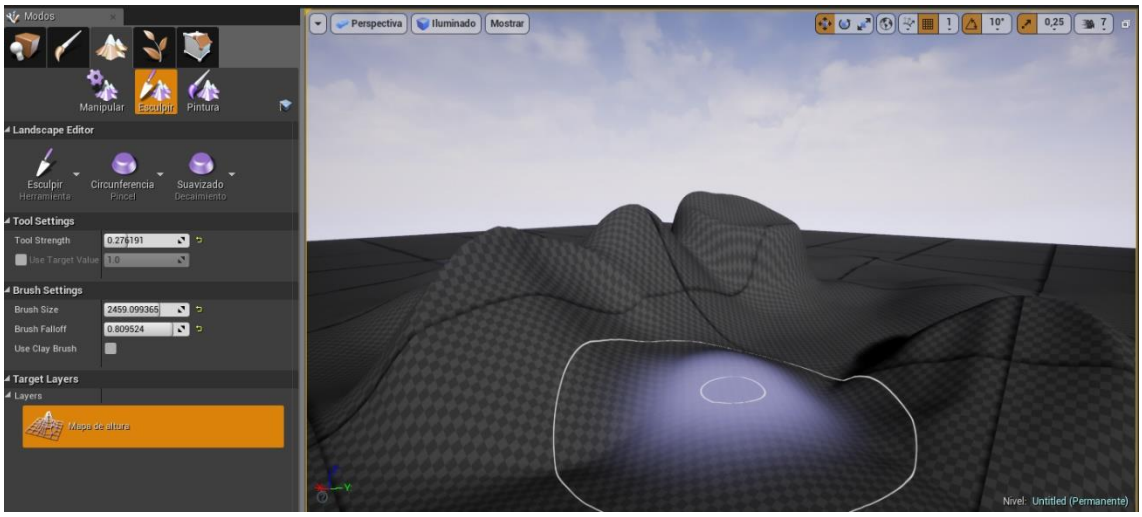

Figura 39: Esculpiendo terreno en UE4

Sin embargo, estas herramientas no poseen mucha precisión, y además el terreno es demasiado grande como para esculpir con detalle. Por esta razón, se decide buscar una manera alternativa de crear un terreno para después importarlo a Unreal.

Se probó la herramienta *Terrain.party* (https://terrain.party), en la que se puede obtener un *Heightmap* de cualquier parte del mundo colocando el cuadrado azul en el área que se desee.

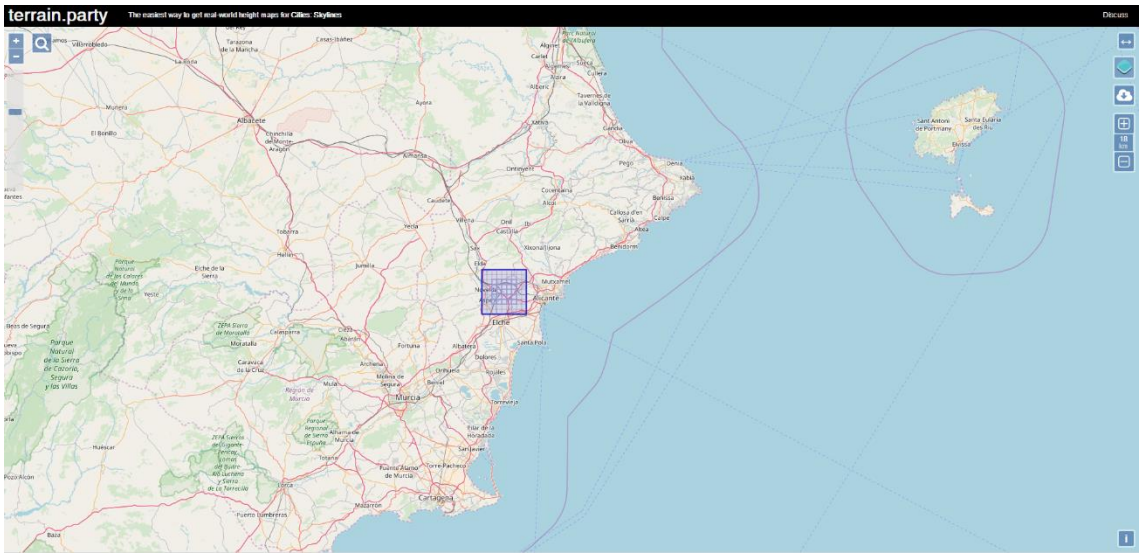

Figura 40: Herramienta Terrain Party

Fuente: https://terrain.party

En Unreal se selecciona *Importar desde el archivo* en el modo *Manipular* de *Terreno* y se escoge la imagen proporcionada por *Terrain.party*. Al igual que en *Crear Nuevo*, se escogen los parámetros de resolución y demás. El problema es que con este método no se tiene control de la forma exacta que va a tener el terreno, y éste puede tener áreas demasiado rugosas, áreas puntiagudas o áreas muy planas.

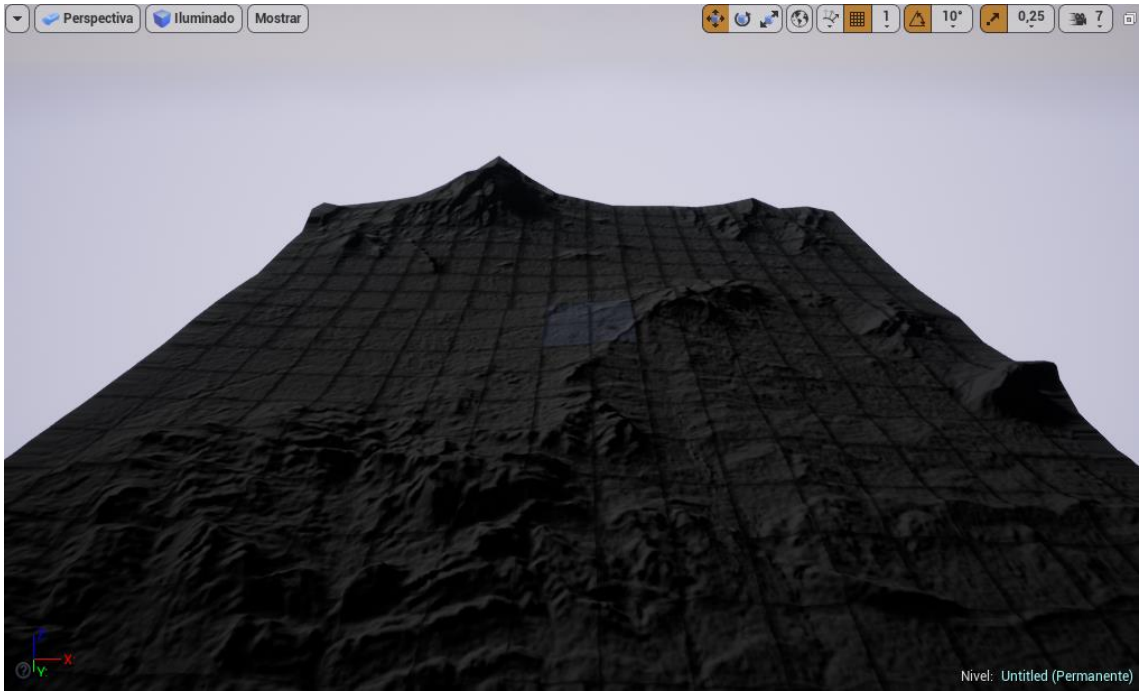

Figura 41: Terreno importado en UE4 mediante un Heighmap obtenido con Terrain.party

Para arreglarlo se podían utilizar las herramientas de Unreal para esculpir sobre este terreno ya creado, pero al igual que antes, vuelve a ser demasiado trabajo para un resultado poco convincente. Es necesario crear el terreno de una manera más controlada y que quede natural.

Esto se pudo obtener utilizando la herramienta de creación de terreno *World Machine*. A partir de una serie de *devices*, se crea la forma básica del terreno: se le añade detalle, erosión, etc.

Nodos o *devices* utilizados:

- *Radial Gradient:* crea formas centradas en un punto.
- *Advanced Perlin Noise:* es un generador de terreno altamente personalizable. Funciona creando y solapando varias capas de ruido.

- *Erosion*: como su nombre indica, simula la erosión natural del terreno.
- *Height Output:* se guarda la salida, que posteriormente se importa en Unreal

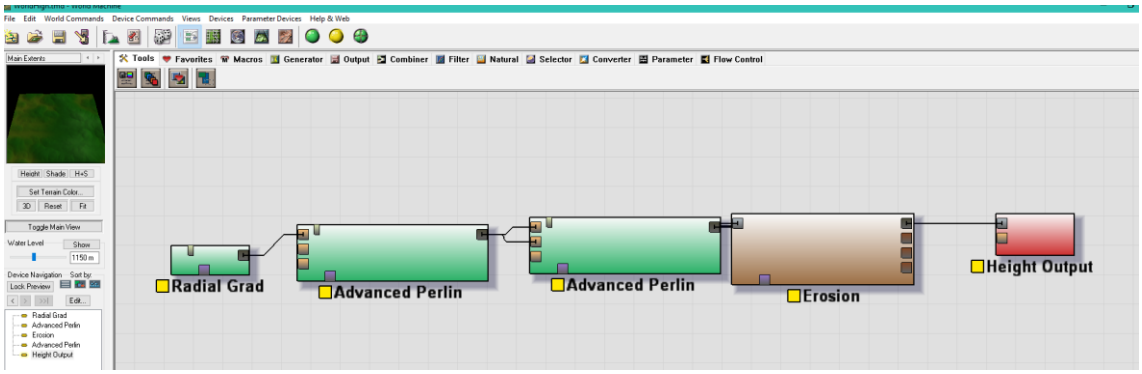

Figura 42: Estructura de *devices* realizada en World Machine

Con esta serie de *devices* se obtiene el siguiente terreno como resultado:

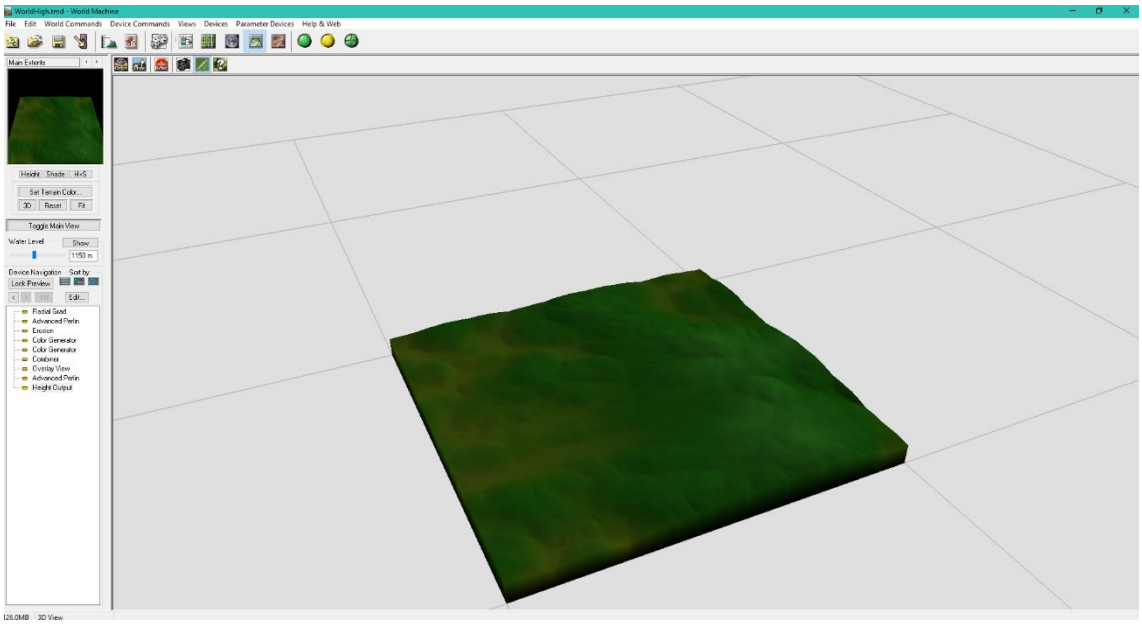

Figura 43: Terreno creado en World Machine

Entrando en *World Commands – Project World Parameters* se escoge la resolución deseada: 505x505, que es una de las recomendadas en la documentación de Unreal.

Con el nodo *Height Output* se exporta con el formato RAW16. En Unreal, se importa del mismo modo que antes, y éste es el resultado:

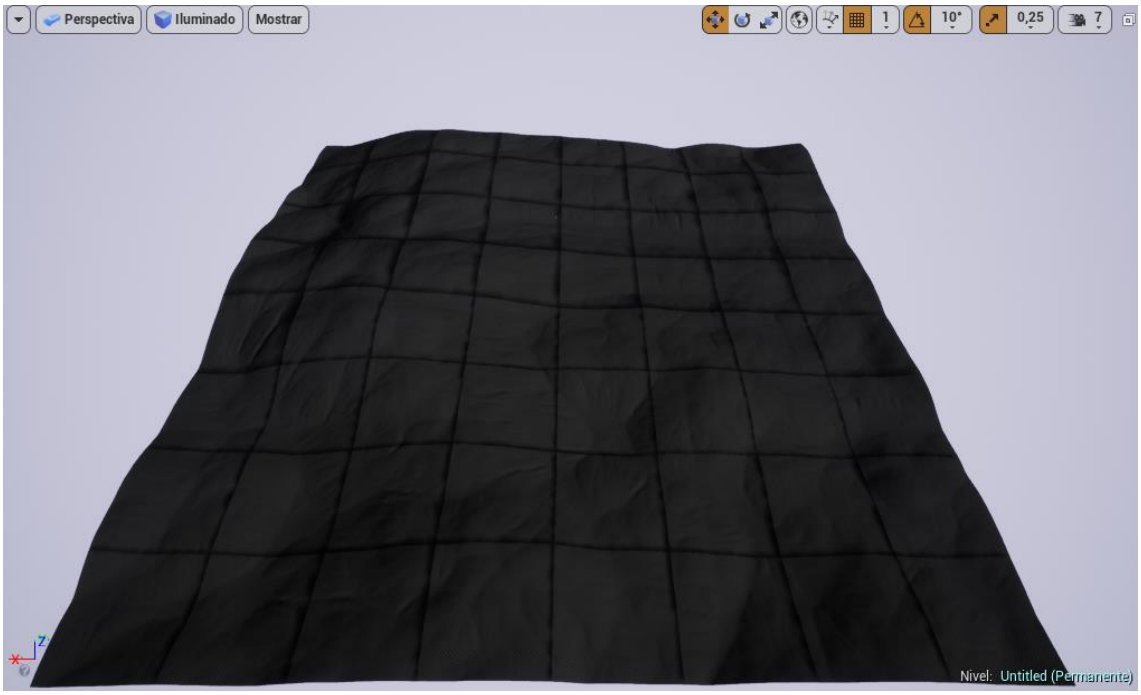

Figura 44: terreno creado en World Machine importado a UE4

Texturizado

Se realiza en Unreal. Es preferible tener varias texturas en el terreno para que no sea demasiado uniforme. El primer material realizado para ello mezcla cinco texturas diferentes y hace uso de mapas normales:

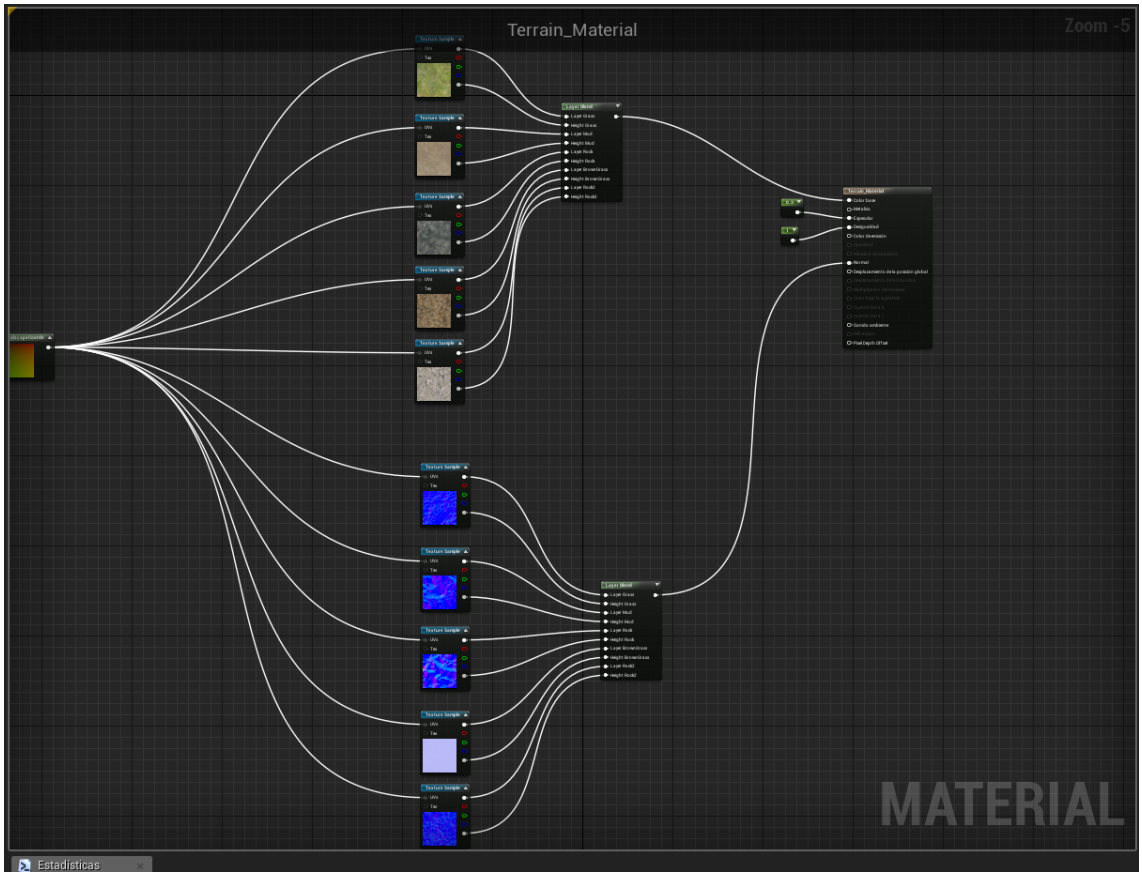

Figura 45: Árbol de nodos del primer material de terreno

Mediante el nodo *Landscape Layer Blend* se sitúa cada textura en una capa, que posteriormente puede seleccionarse en el modo *Pintura*. Se utiliza este nodo dos veces: tanto en los mapas de color como en los de normales. Así se pueden pintar diferentes texturas sobre el terreno.

Además, se utiliza el nodo *Landscape Layer Coords*, que genera las coordenadas UV necesarias para poder aplicar los materiales al terreno. En este nodo se puede elegir la escala del mapeado, que por defecto vale 0. Se aumenta este valor para tener el tamaño de textura correcto.

Finalmente, una vez terminado el material, éste se asigna al objeto del terreno. En el modo Pintura aparecen las diferentes texturas (*layers*). Para cada una, se le crea una *capa de peso combinado*. Tras esto, ya se puede aplicar la pintura seleccionando la que se desee.

Figura 46: Terreno con el primer material creado; modo Pintura (visto de lejos)

Figura 47: Terreno con el primer material creado (visto de cerca: dos texturas mezclándose)

El problema del material creado es que, como se ve en la figura 46, produce efecto de *tiling* al alejarse del suelo. Para solucionarlo sería necesario añadir más texturas y pintar manualmente todo el terreno con el pincel en tamaño muy pequeño, algo poco eficiente.

Después de varias pruebas, se llega al siguiente material, que combina cuatro tipos de textura:

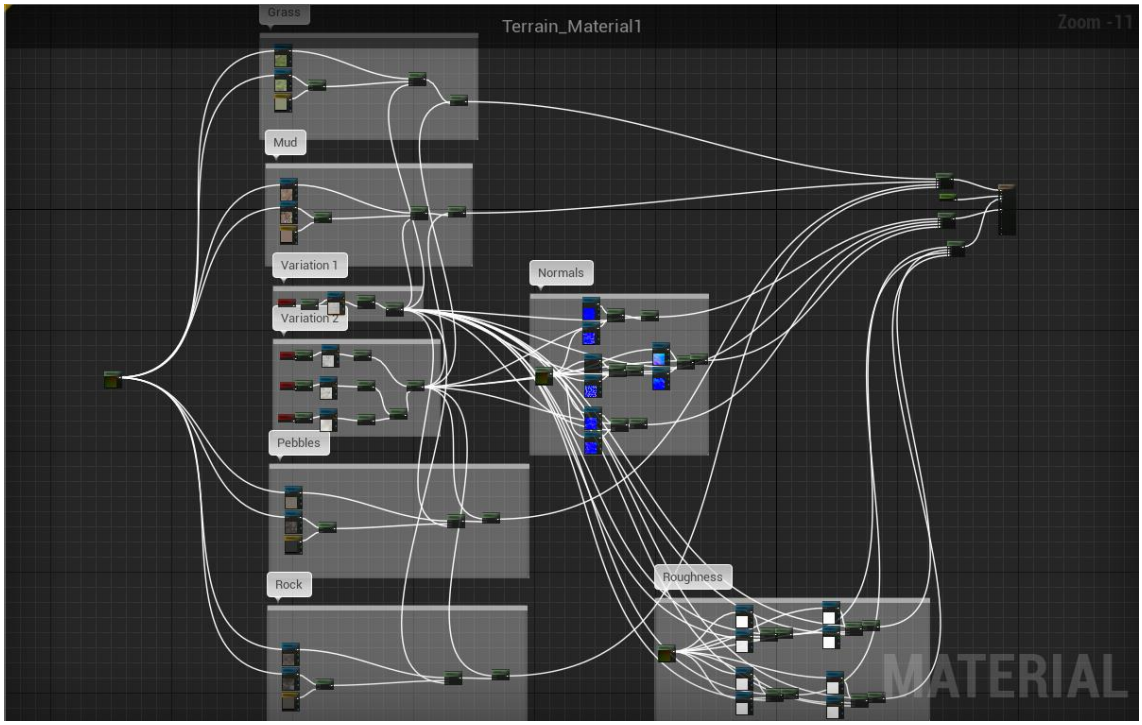

Figura 48: Árbol de nodos del material final del terreno

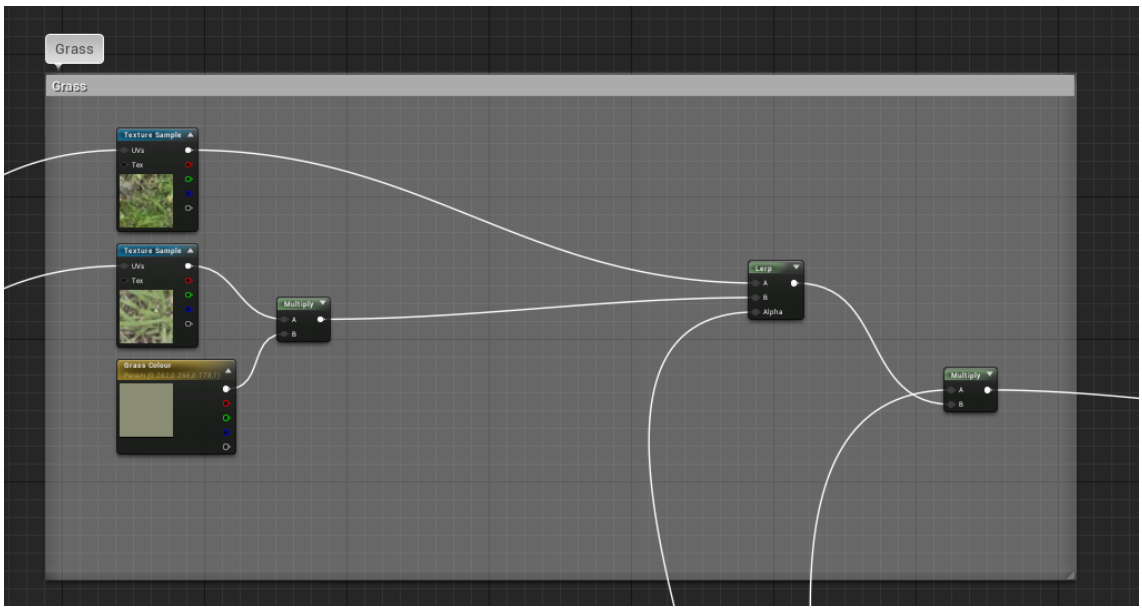

Figura 49: Parte de la textura de hierba en el material final del terreno

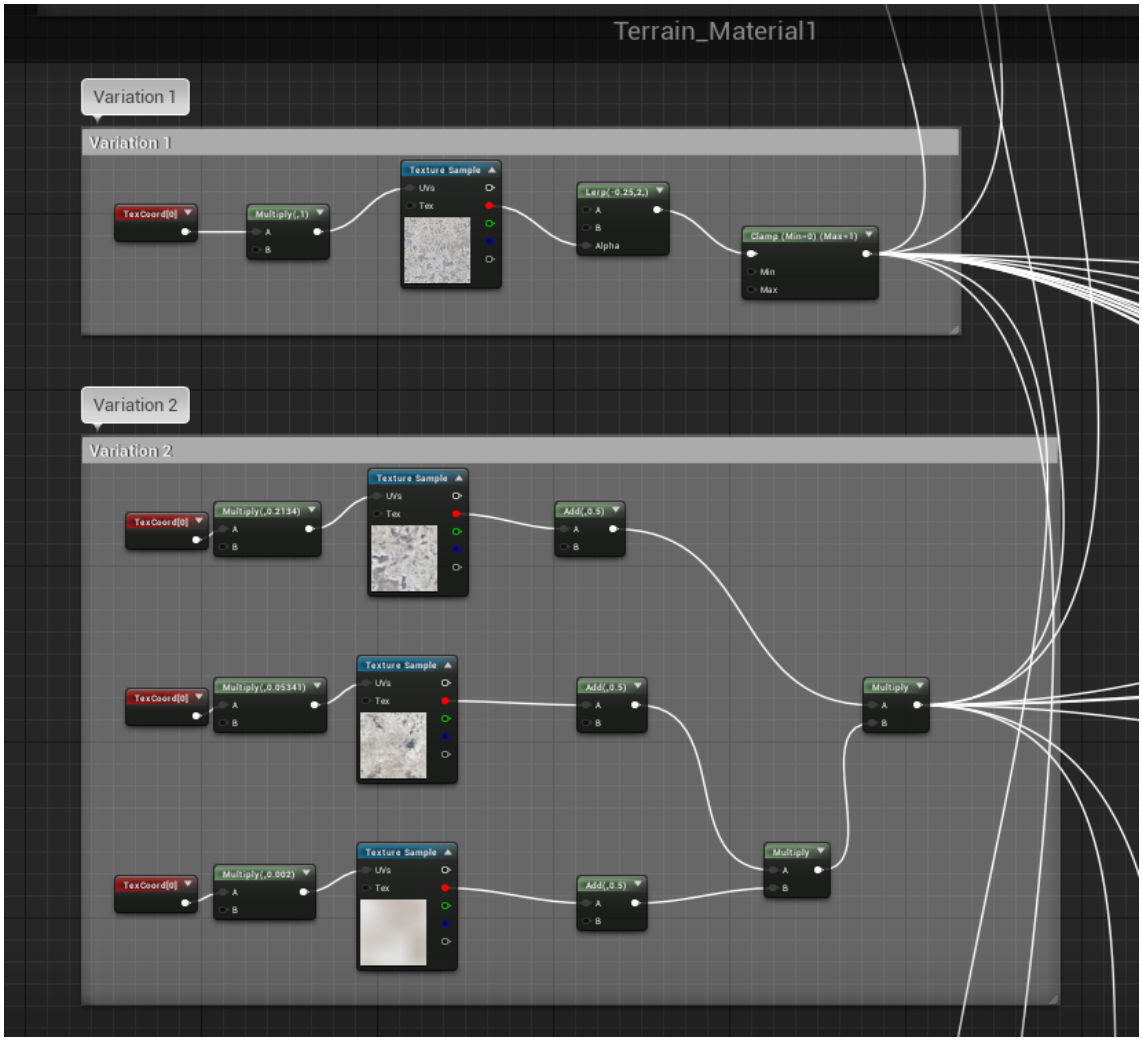

Figura 50: Parte de la textura de variación en el material final del terreno

Para cada tipo –hierba, barro, guijarros y roca– se mezclan dos texturas y un color similar a ellas (Figura 49). También se añade variación con una textura proporcionada en *Kitedemo (T_Macro_Variation)*. El uso de esta textura se repite hasta un total de 4 veces (como se puede ver en la figura 50), cambiando para cada una las coordenadas y así utilizar diferentes áreas de la misma. Se utilizan nodos *Multiply*, *Linear Interpolate*, *Add* y *Clamp* para combinar las diferentes texturas, y como antes, *Landscape Layer Blend* para las cuatro capas, y *Landscape Layer Coords* para las coordenadas UV.

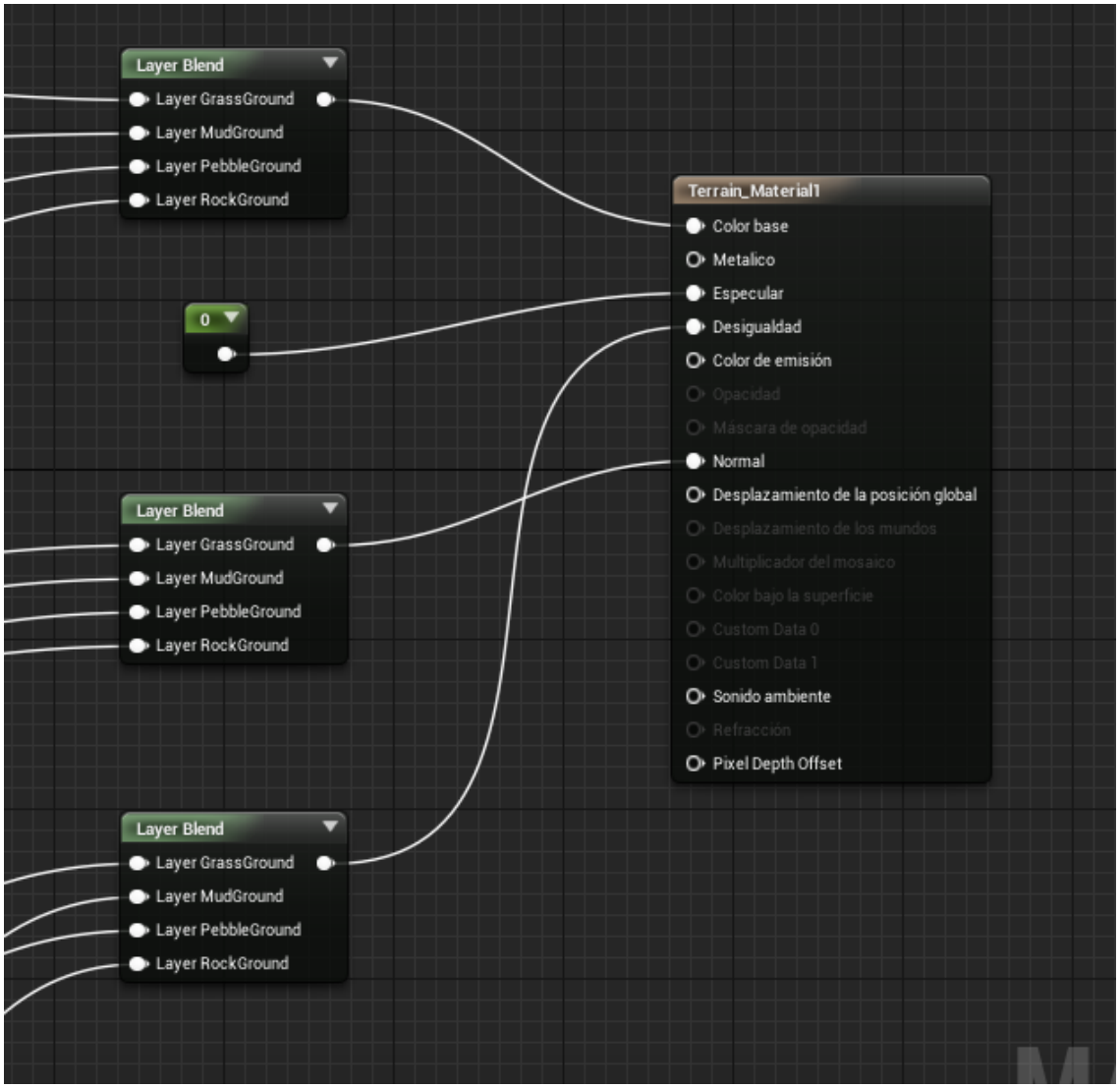

Figura 51: Parte de los nodos Landscape Layer Blend y nodo principal

Los mapas de normales y de desigualdad se combinan a su vez sin utilizar el color, como muestra la siguiente imagen:

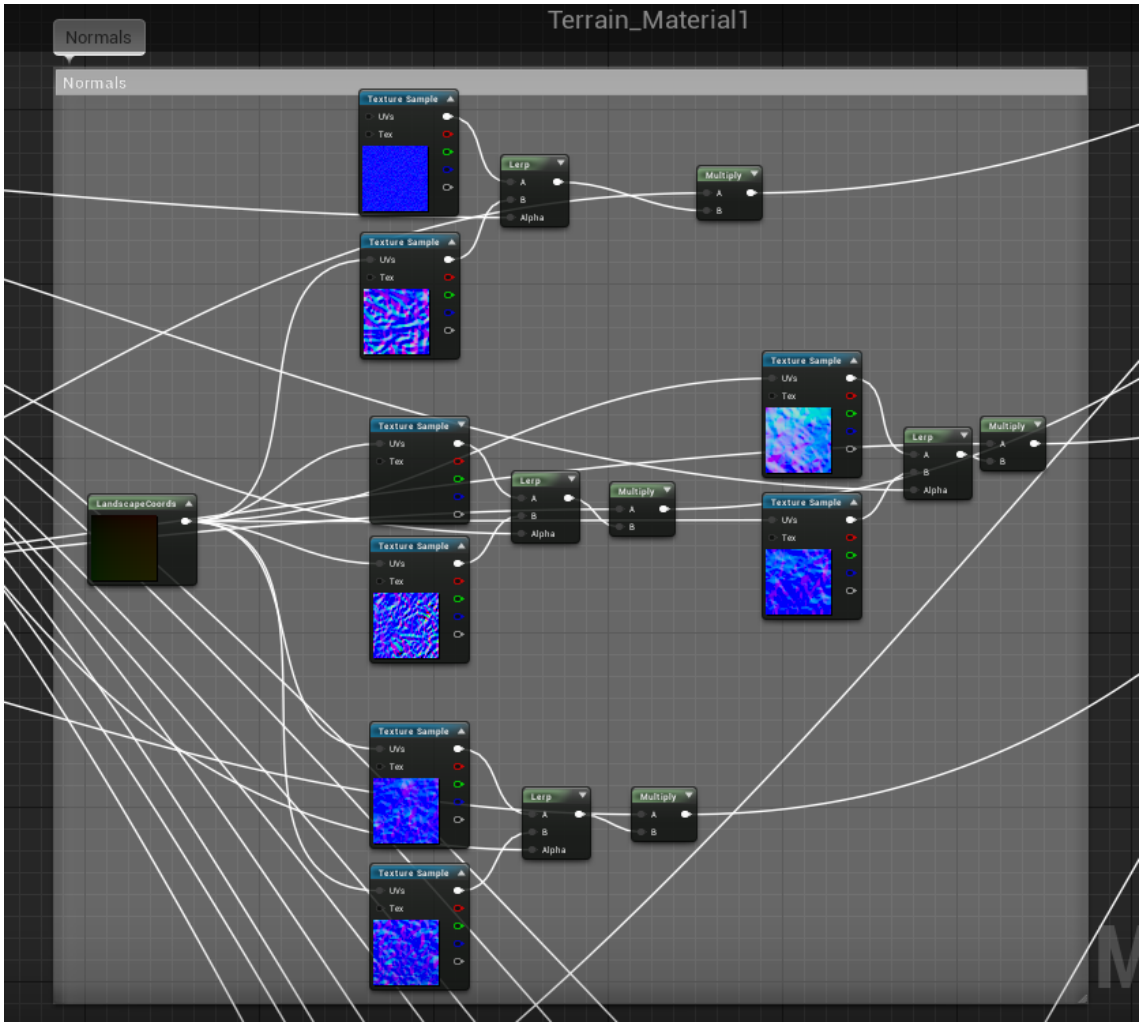

Figura 52: Parte de los mapas normales en el material final del terreno

El material resultante crea variaciones en el color, de modo que se disimula en gran medida el efecto de *tiling*, especialmente con las texturas de hierba. La mayor parte del terreno se pintará utilizando la capa de hierba.

Figura 53: Terreno con el material final, visto de lejos

En los *Texture* Sample, para este material concreto, al combinar texturas diferentes es necesario cambiar el valor de *Sampler Source* que viene por defecto a *Shared: Wrap*, para que compartan la misma muestra de textura y así no tener un número demasiado alto. Con el valor por defecto, *From Texture Asset*, al pìntar teniendo demasiadas texturas aparecen secciones "grises", sin textura, como se puede apreciar en la Figura 55.

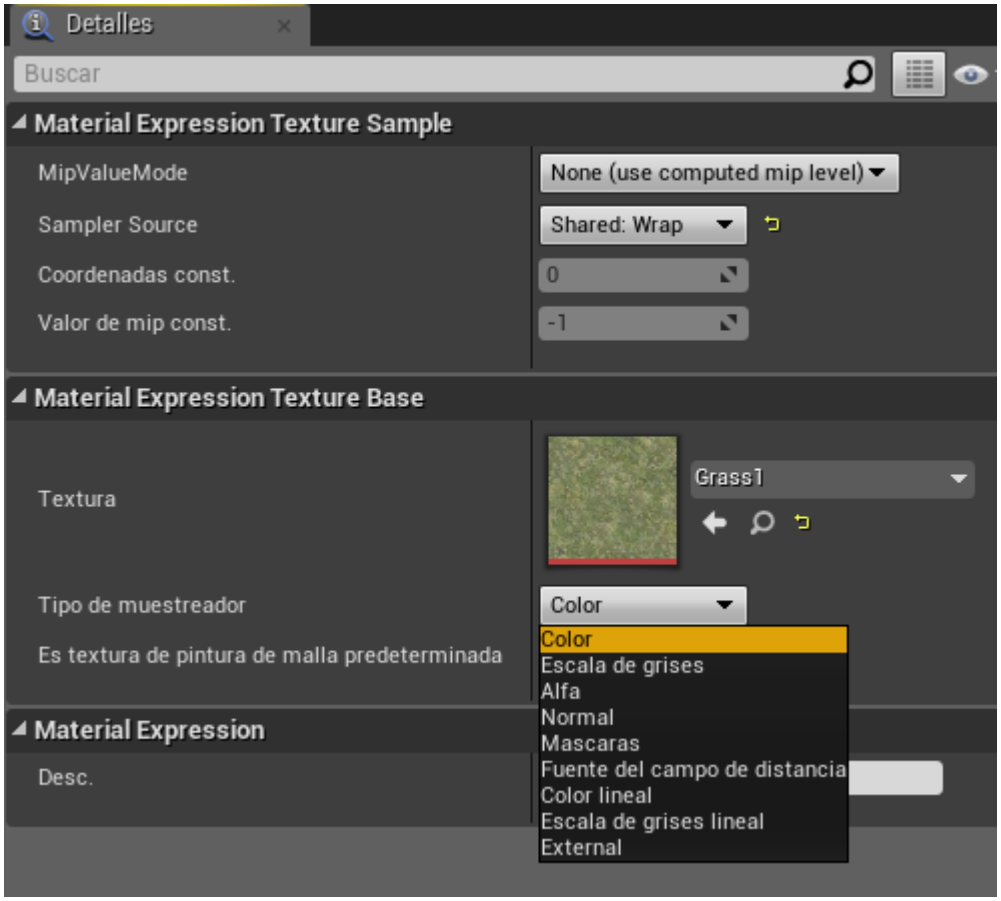

Figura 54: Detalles de nodo *Texture Sample*

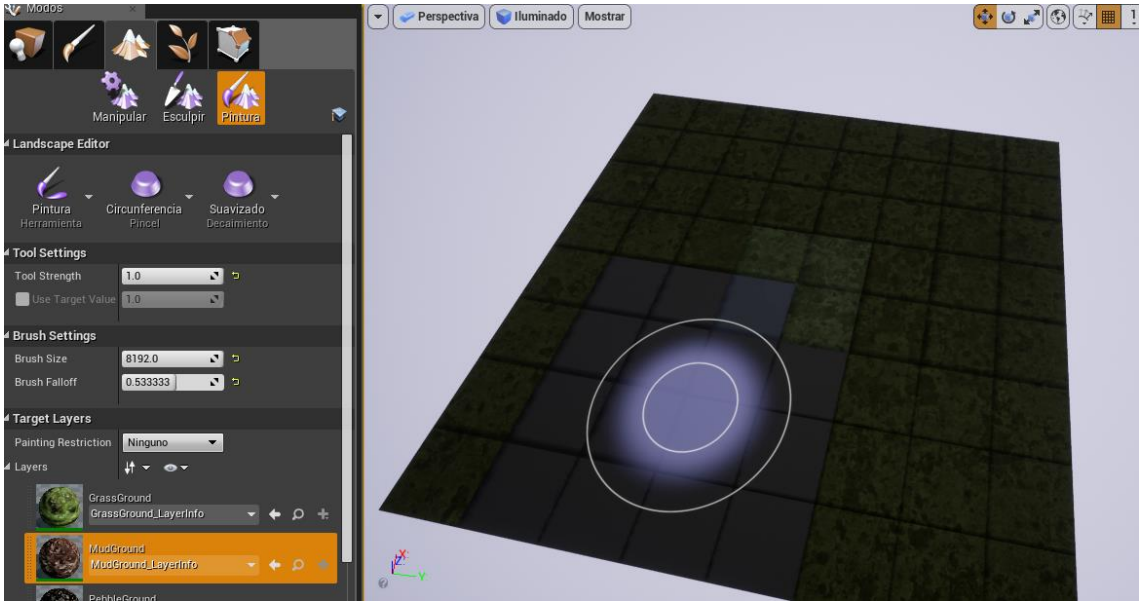

Figura 55: Problemas al pintar cuando no se selecciona *Shared: Wrap*

5.4.2. Montañas

Modelado

Para el caso de las montañas, en Blender se activa en *User preferences* el addon *Add Mesh: A.N.T.Landscape*. Tras esto, al hacer Shift + A en la vista 3D, entre las diferentes mallas que se pueden seleccionar aparecerá *Landscape*.

Al seleccionar este tipo de malla, se crea una montaña por defecto. En el panel de la izquierda aparecerán las opciones del addon para modificar la malla.

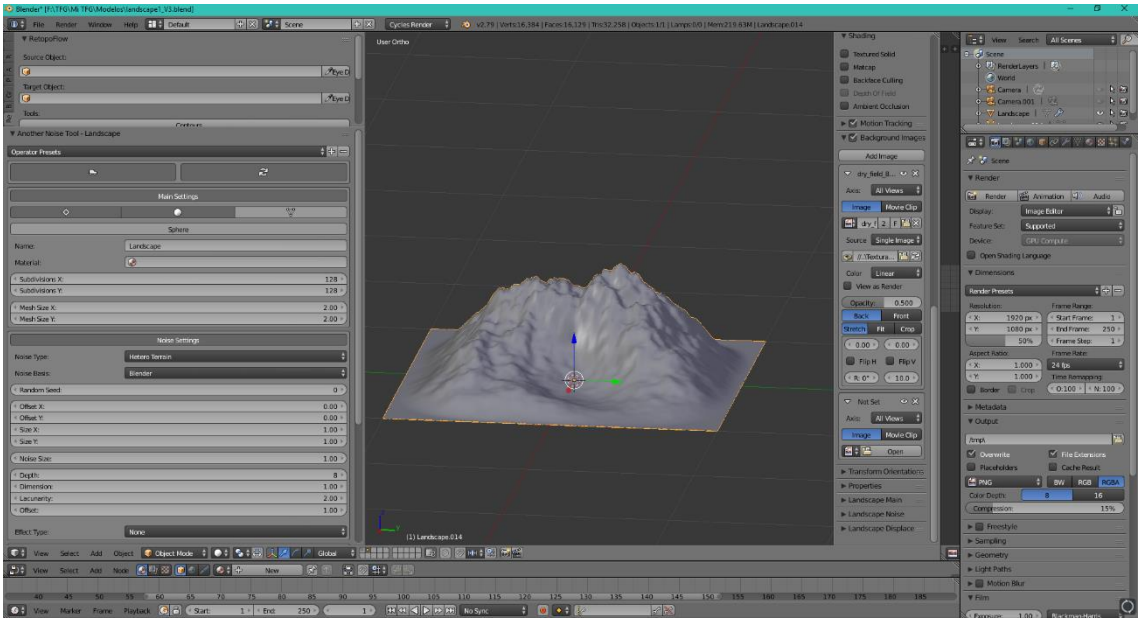

Figura 56: Addon *Add Mesh: A.N.T.Landscape*

Cambiando parámetros como el tipo y tamaño de ruido o el desplazamiento se obtienen diferentes montañas.

Texturizado

En un principio se crea la textura en el propio Blender mediante nodos, añadiéndole desplazamiento, una textura de rocas y dos texturas de nieve –cada una con sus mapas de *albedo*, *height*, *roughness* y *specula* –.

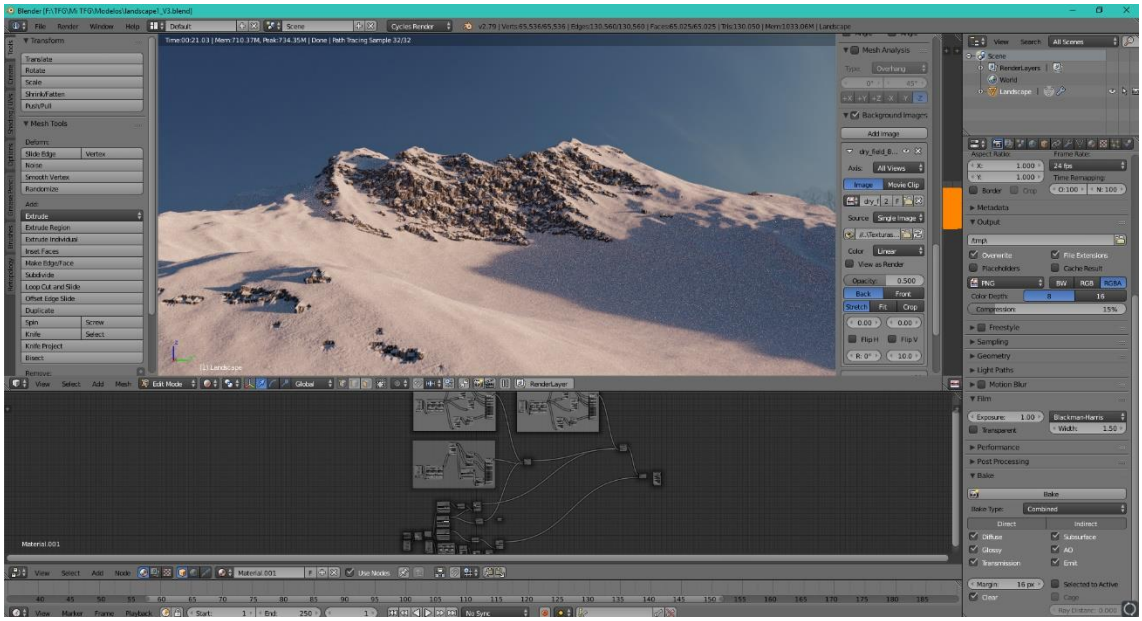

Figura 57: Primera montaña realizada

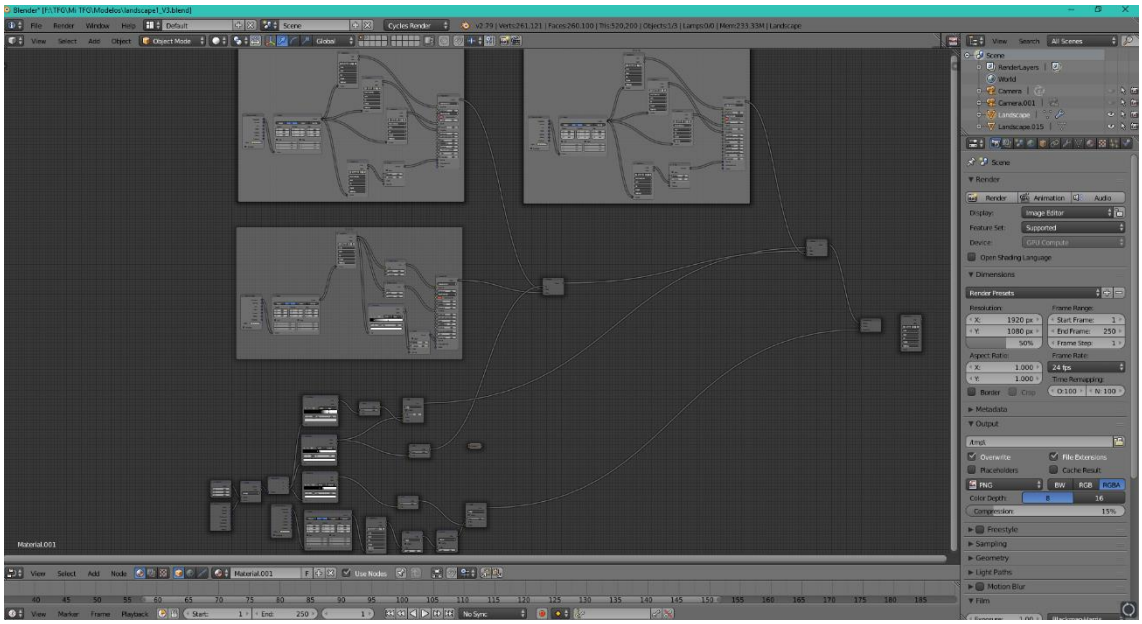

Figura 58: Árbol de nodos de la primera montaña realizada

Figura 59: Render de la primera montaña realizada

A pesar del buen resultado en Blender, al importar el modelo y la textura a Unreal no era tan bueno al mirarse desde muy cerca, ya que las imágenes de textura perdían nitidez al estar aumentadas. Debido a ello, finalmente se descarta esta forma de texturizar para elementos tan grandes como las montañas.

Figura 60: Primera montaña realizada, vista de lejos en Unreal

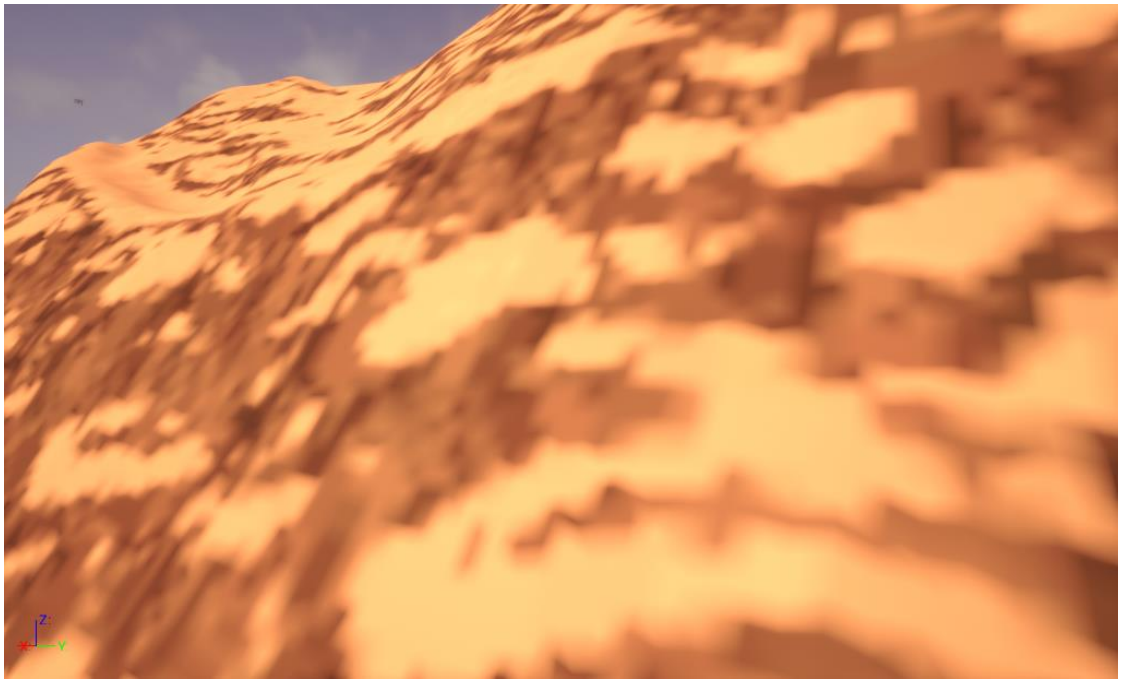

Figura 61: Primera montaña realizada, vista de cerca en Unreal

El siguiente método que se probó para texturizar este tipo de mallas, fue utilizando el modo *Texture Paint* de Blender. Se crea una brocha a la que se le añade una de las texturas, y se va colocando sobre el objeto: en este caso, la montaña. Mediante el uso de diferentes brochas –es decir, diferentes imágenes de textura– y jugando con la opacidad de éstas se consigue cierta variación en la textura final de la montaña, y así no se ve repetitiva.

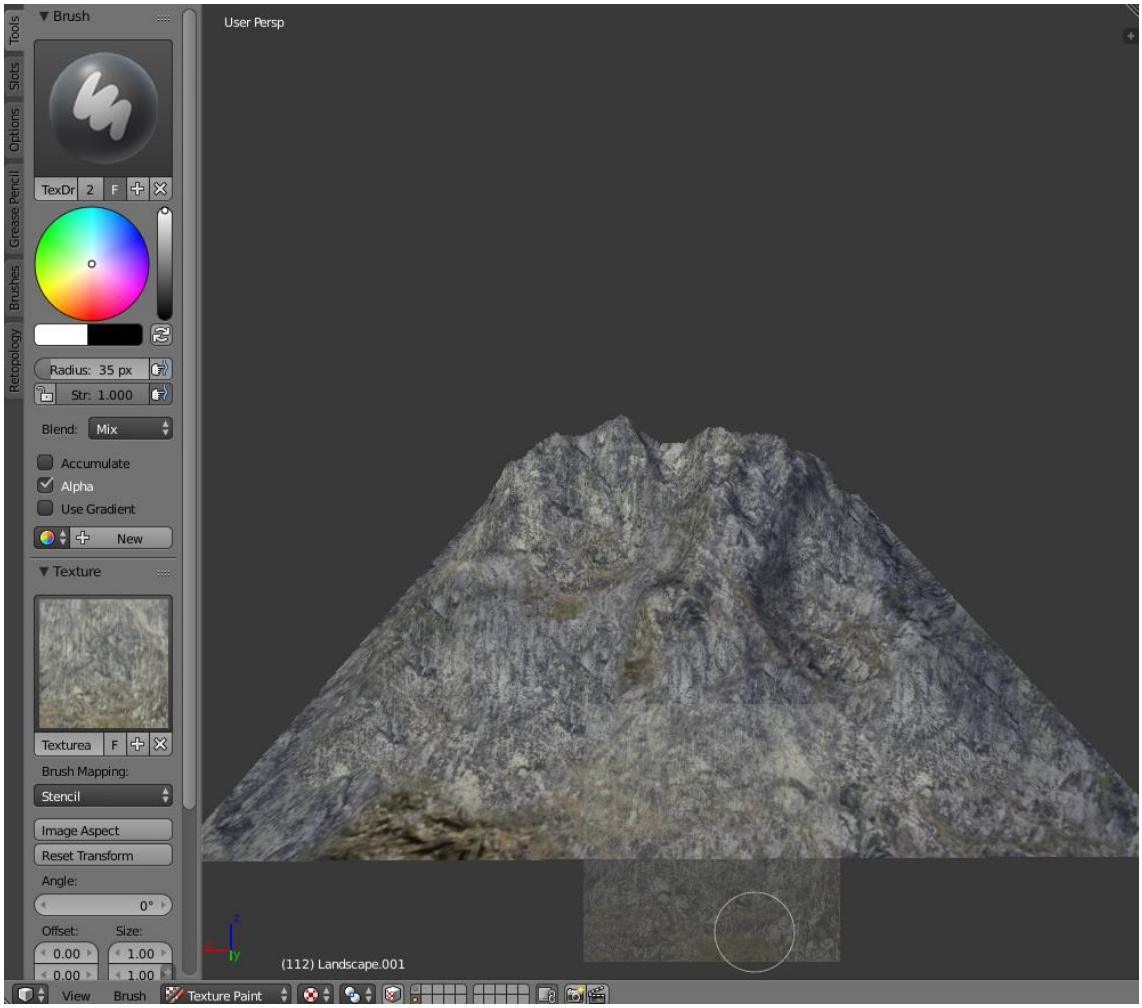

Figura 62: Montaña en modo *Texture Paint*

Sin embargo, seguía existiendo el mismo problema que antes. Aunque ahora se pueda hacer *zoom* para volver a pintar sobre una parte difuminada, las montañas son mallas excesivamente grandes para texturizar manualmente de forma tan reducida.

De lejos se puede conseguir un buen resultado, como muestra la siguiente imagen:

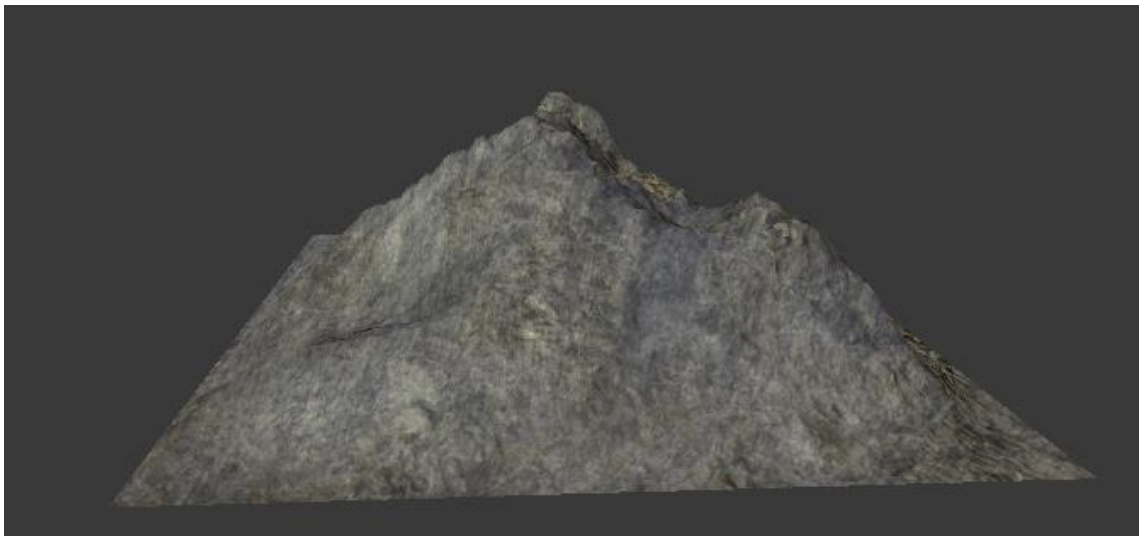

Figura 63: Montaña texturizada en *Texture Paint*, vista de lejos

Por otro lado, si se mira desde cerca, se ve demasiado borrosa:

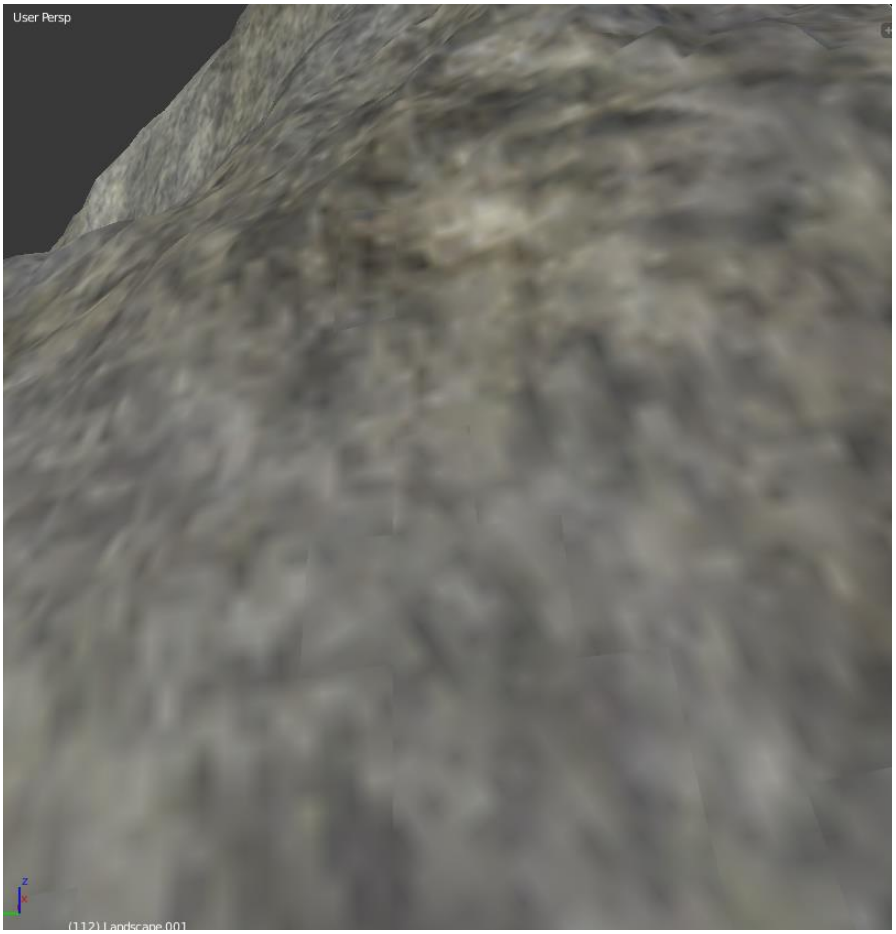

Figura 64: Misma montaña, vista de cerca

También se ha probado a combinar dos imágenes de montaña de manera más sencilla utilizando una textura de ruido y variando la escala del mismo (al igual que en 5.4.5. Rocas con musgo). Aunque así se puede conseguir buena textura de cerca en toda la montaña, esto se traducía en un resultado malo al alejarse de ella, haciendo evidente el *tileado* (repetición del patrón) y/o quedando con un color muy uniforme, con poco detalle.

Figura 65: Montaña texturizada mediante nodos, vista de cerca

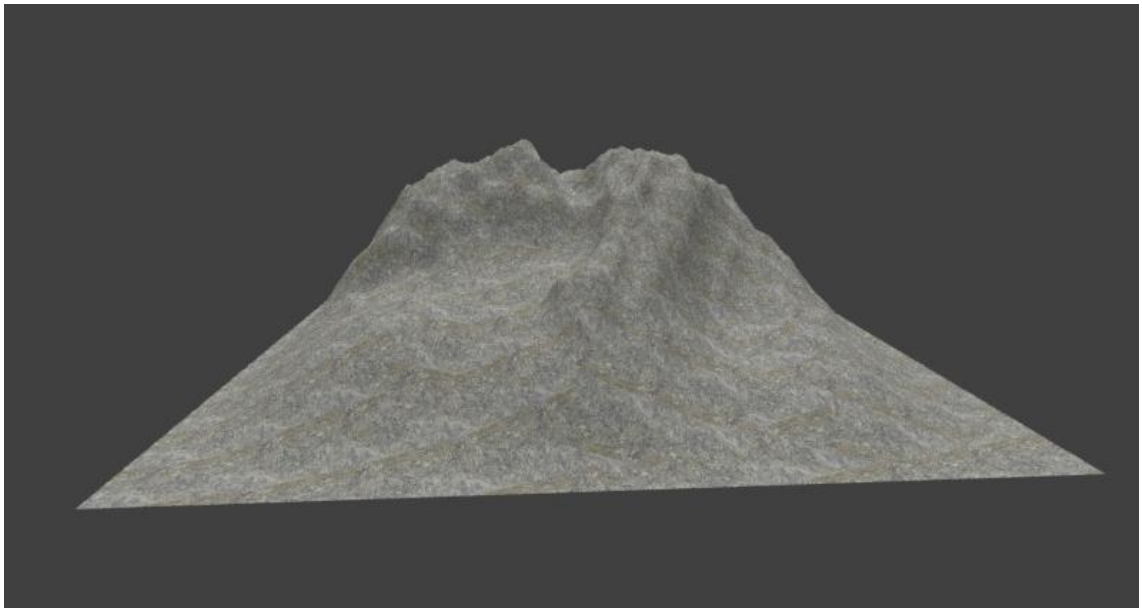

Figura 66: misma montaña, vista de lejos

En conclusión, la textura de las montañas no tenía buen resultado bien de cerca, o bien de lejos, por lo que se busca una alternativa: Substance Painter, herramienta con la que finalmente se texturizan.

5.4.3. Hojas caídas

Modelado

En Blender se activa el addon *Import-Export: Import Images as Planes*. Con Shift + A – *Mesh* – *Images as Planes*, se abre el explorador de archivos y se selecciona la imagen (en este caso, de una hoja) a importar. Aparecerá un plano del mismo tamaño que la imagen, con ésta como textura.

Las imágenes importadas tienen formato *.png* y transparencia alrededor de la hoja. Por defecto aparecerán con el siguiente material:

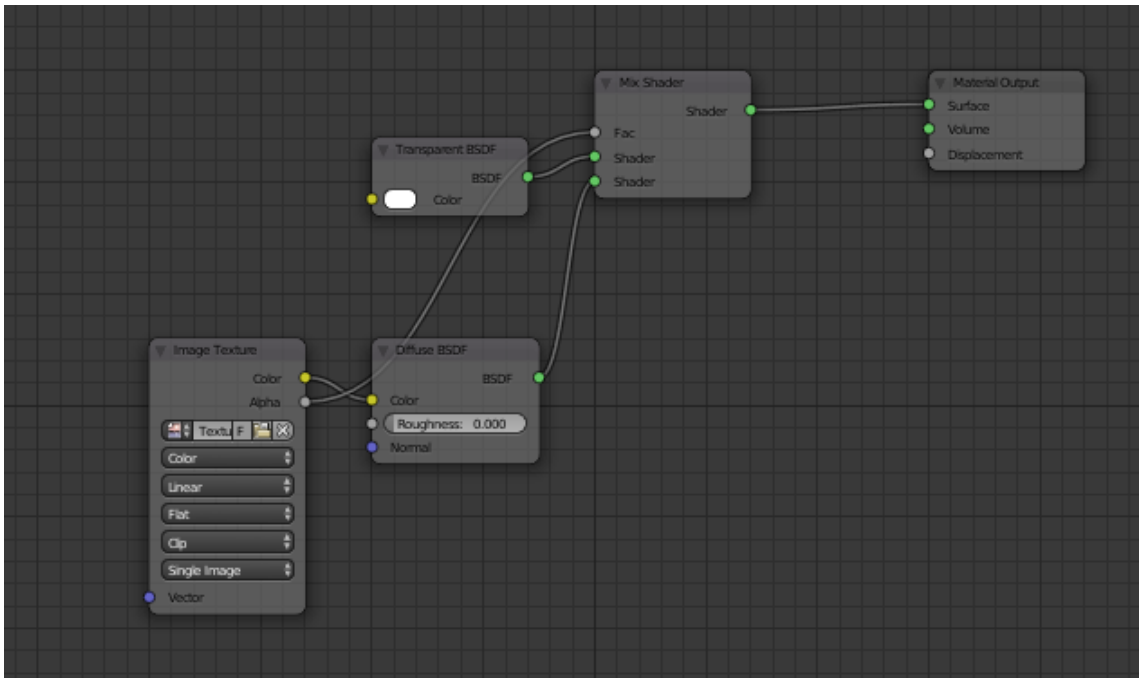

Figura 67: Árbol de nodos del material por defecto de *Images as Planes*

En *Image Texture* se ha colocado la imagen importada y se utiliza como color del *Diffuse BSDF*.

El nodo *Transparent BSDF* proporciona la transparencia deseada cuando se combina con el *Diffuse BSDF*. Para ello se utiliza un nodo *Mix Shader*, y el peso del promedio queda determinado por un factor: en este caso, el canal alfa de la imagen.

En *Edit Mode* se deforma un poco el plano para que la hoja no se vea completamente lisa sobre el suelo. Para ello mediante Ctrl + R se crean nuevos segmentos

en el plano. Utilizando la rueda del plano se determina la cantidad. Tras esto se desplazan algunos vértices en el eje Z.

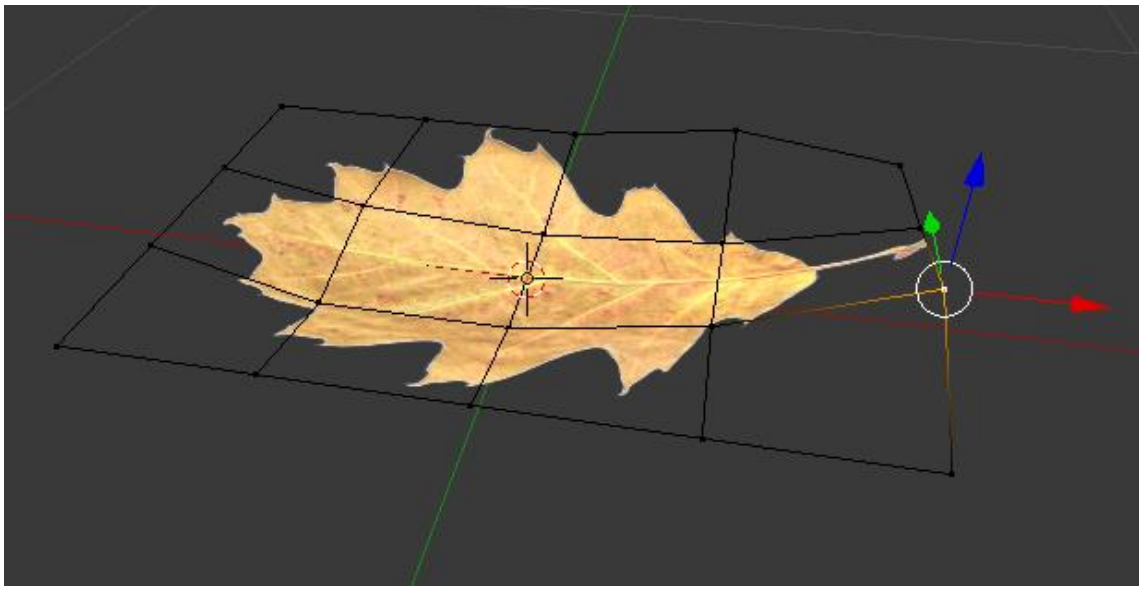

Figura 68: Deformación del plano de la hoja en *Edit Mode*

Texturizado

Al material ya creado se le añaden nodos de *Bump*, *Subsurface Scattering*, *Glossy BSDF* y *Translucent BSDF*. Se combinan con nodos *Mix Shader* y *Add Shader*.

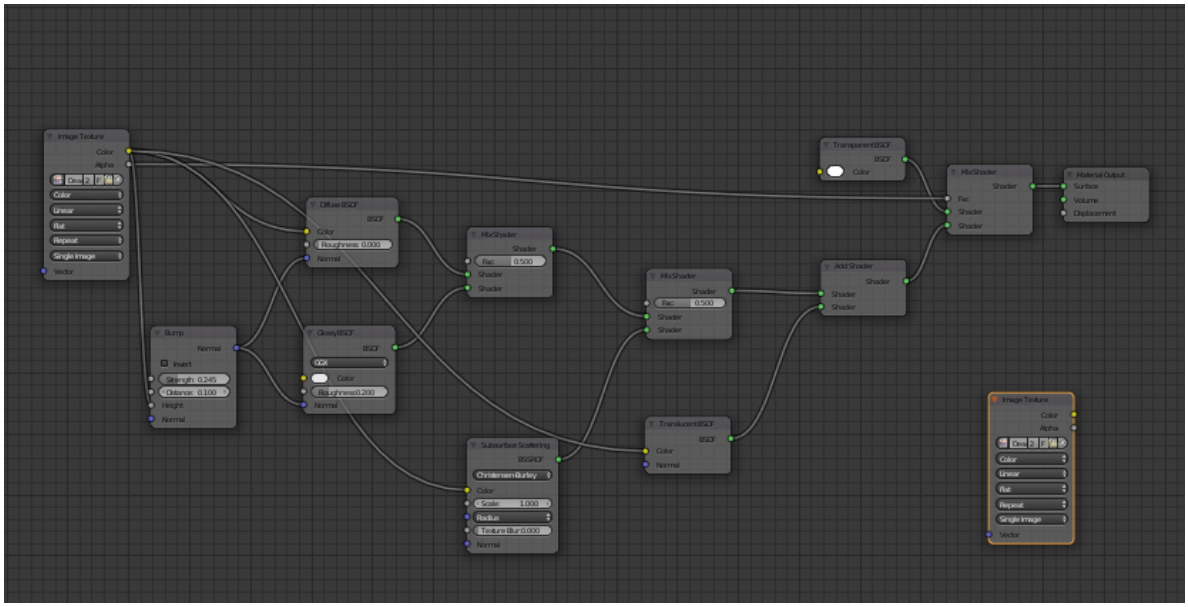

Figura 69: Árbol de nodos creado para las hojas

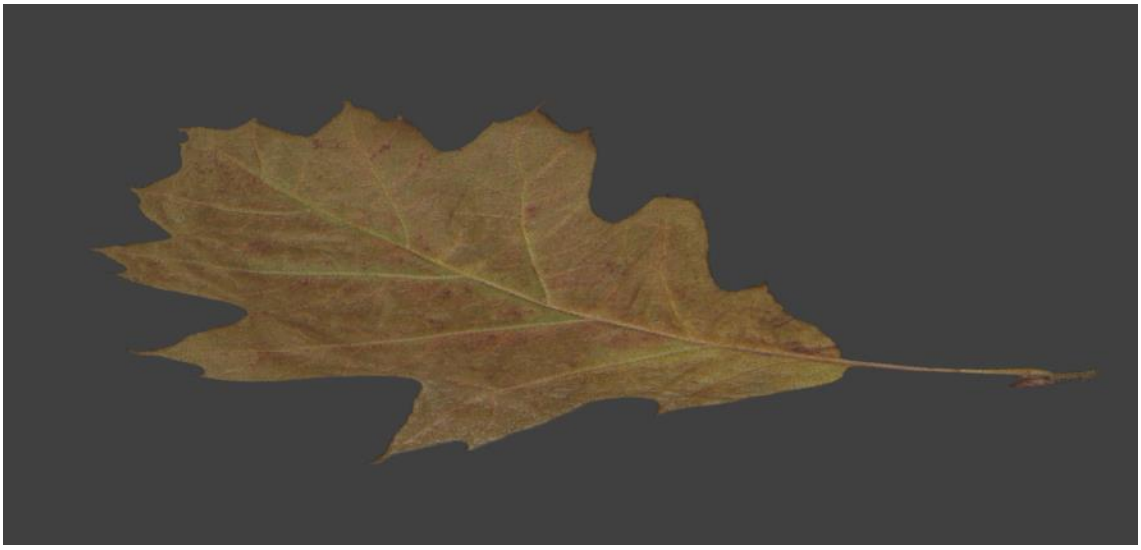

Figura 70: Hoja en *Render Preview*

El mapa normal se obtiene mediante NormalMap Online. El mapa de opacidad se crea en Photoshop. Para los mapas de *Translucency* y *Roughness* se utiliza el método *Bake*.

En Unreal se crea una instancia de *Foliage_Master_Mat* con los mapas de textura de la hoja. Los parámetros de viento se ponen a 0 para este caso.

Tienen un total de tres niveles de detalle, creados de manera automática, teniendo el último nivel únicamente dos triángulos.

Finalmente, se añade a *Foliage* para añadir de manera sencilla numerosas instancias del objeto.

5.4.4. Rocas

Modelado

En un principio se hacen a partir de Shift A – *Mesh* – *Ico Sphere*, y deformando la malla en *Sculpt Mode* con la brocha *Sculpt Draw*, que mueve los vértices hacia afuera o adentro (si se mantiene pulsado Ctrl) y la brocha *Smooth* para eliminar irregularidades. Luego, con el modificador *Displace* se añade algo de variación a la superficie.

Finalmente se selecciona *Smooth* como método de sombreado.

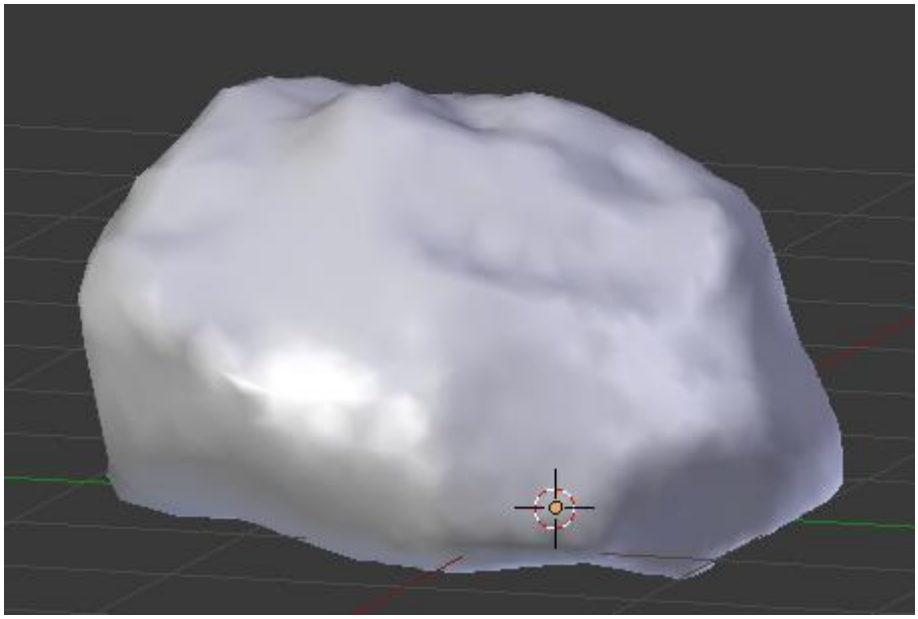

Figura 71: Una de las primeras rocas creadas

El resultado se veía muy suavizado, algo que no encajaba para las rocas más grandes, en las que se prefería un efecto más erosionado.

Se buscaron sets de brochas para esculpir en Blender y se modelan las siguientes rocas con ellas.

Se cambia la malla inicial: ahora se empieza a partir de *UV Sphere*, se le añade el modificador *Subdivision Surface* para que tenga una gran cantidad de vértices que permitan moldear con mucho detalle.

Los nuevos sets de brochas tienen algunas que ayudan a crear una superficie más rocosa. Entre ellas hay dos brochas para el suavizado que no estropean tanto la forma como la llamada *Smooth*, que viene por defecto.

Entre las herramientas de *Sculpt Mode* se encuentra la de topología dinámica (*Dynotopo*), que se activa para modificar la malla con más detalle. El tamaño de éste se especifica en píxeles; se le da un valor pequeño.

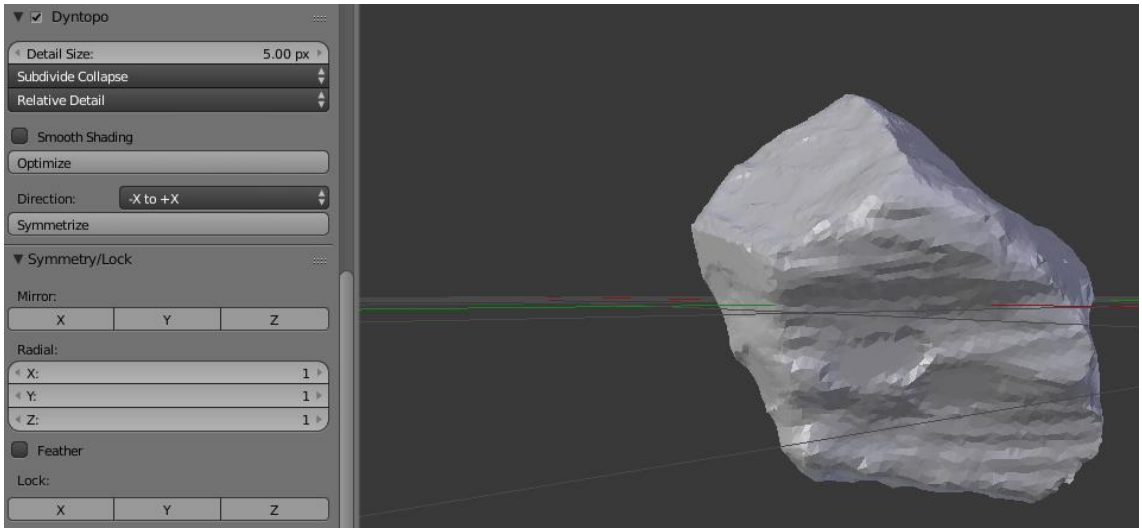

Figura 72: Roca en *Sculpt Mode* con *Dynotopo* activada

Después se selecciona de nuevo el sombreado *Smooth,* y éste es el resultado:

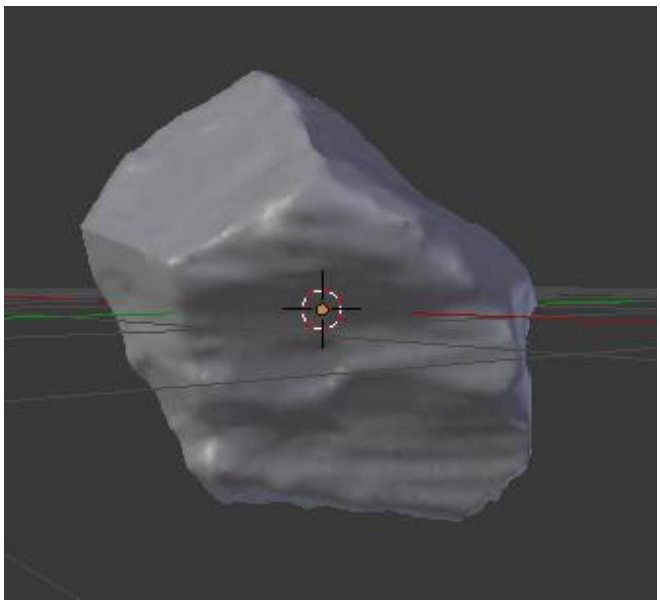

Figura 73: Roca modelada

Al tratarse de una malla con muchos polígonos, es necesario crear otra malla con muchos menos para no cargar tanto el entorno en Unreal.

Se duplica la malla y a la copia se añade el modificador *Decimate*. Se reduce la cantidad de polígonos lo máximo posible manteniendo la forma de la roca.

Así, de cada roca hay una versión de la malla *High Poly* y una *Low Poly*.

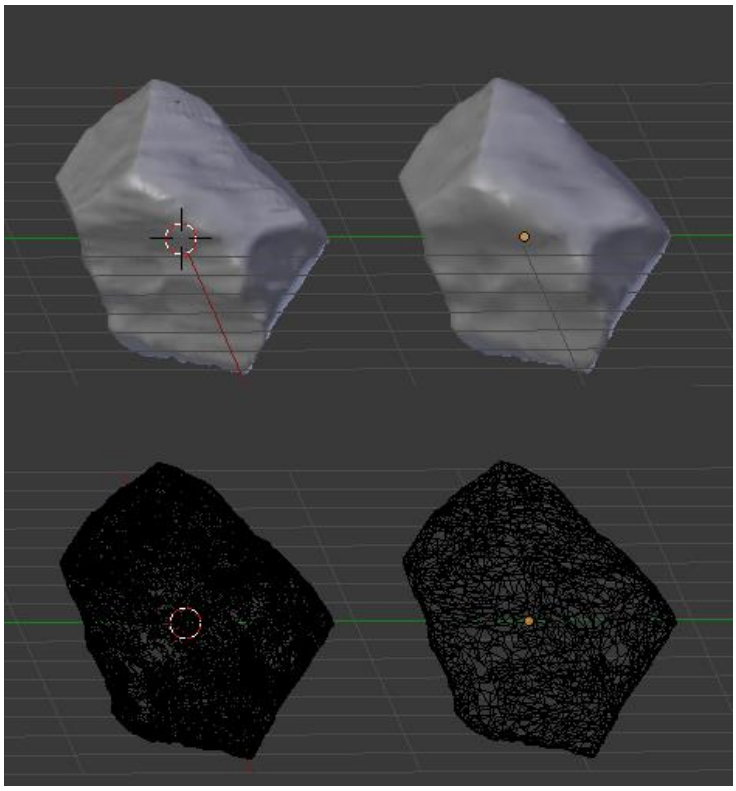

Figura 74: *High Poly* (izquierda) –36.212 triángulos– y *Low Poly* (derecha) –3.620 triángulos– de la misma roca vista en *Viewport Shading Solid* (arriba) y *Wireframe* (abajo)

Texturizado

Así como el primer modelado no era convincente para las rocas más grandes, la textura tampoco lo era, puesto que provocaba el mismo problema que con las texturas de las montañas. Así que, del mismo modo, se utiliza con ellas Substance Painter.

Es posible obtener el mapa normal en Blender a partir de ambas mallas. Éstas se superponen y se utiliza *Bake* seleccionando primero la *High Poly* y después la *Low Poly*, mediante la opción *Selected to Active* y un valor de *Ray Distance* adecuado para que no haya áreas erróneas (por ejemplo, zonas con color naranja o verde oscuro) en el mapa normal.

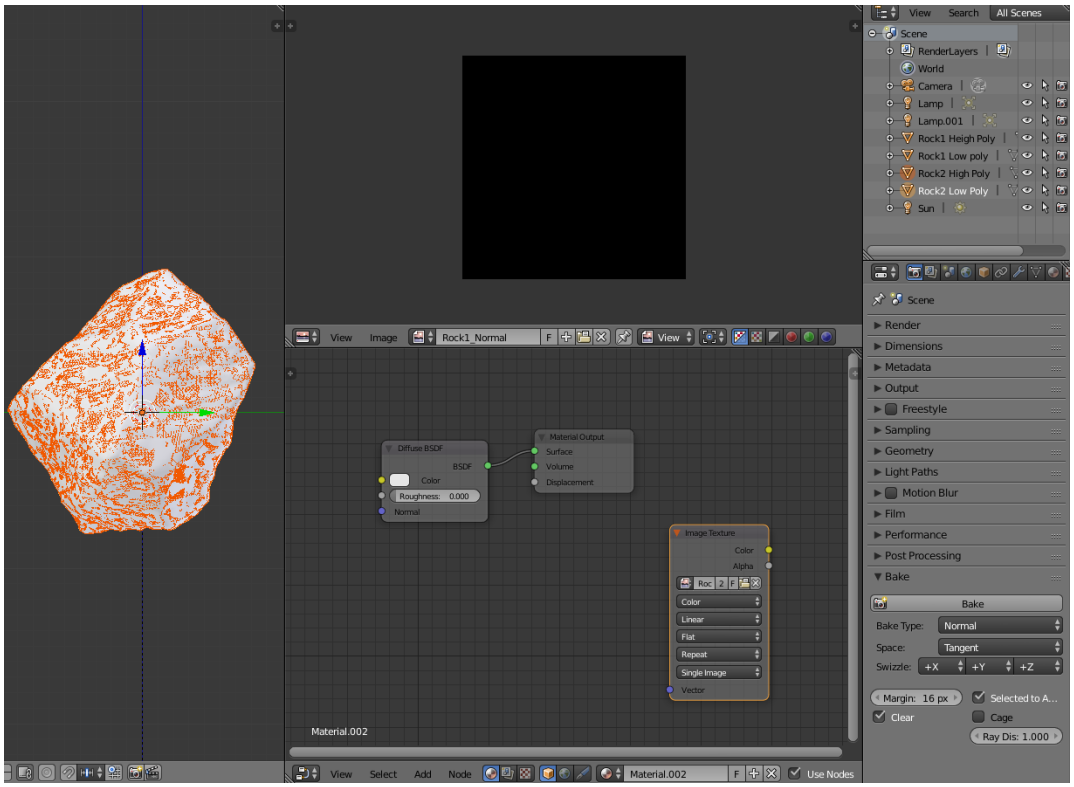

Figura 75: Bake para obtener el mapa normal con ambas mallas superpuestas

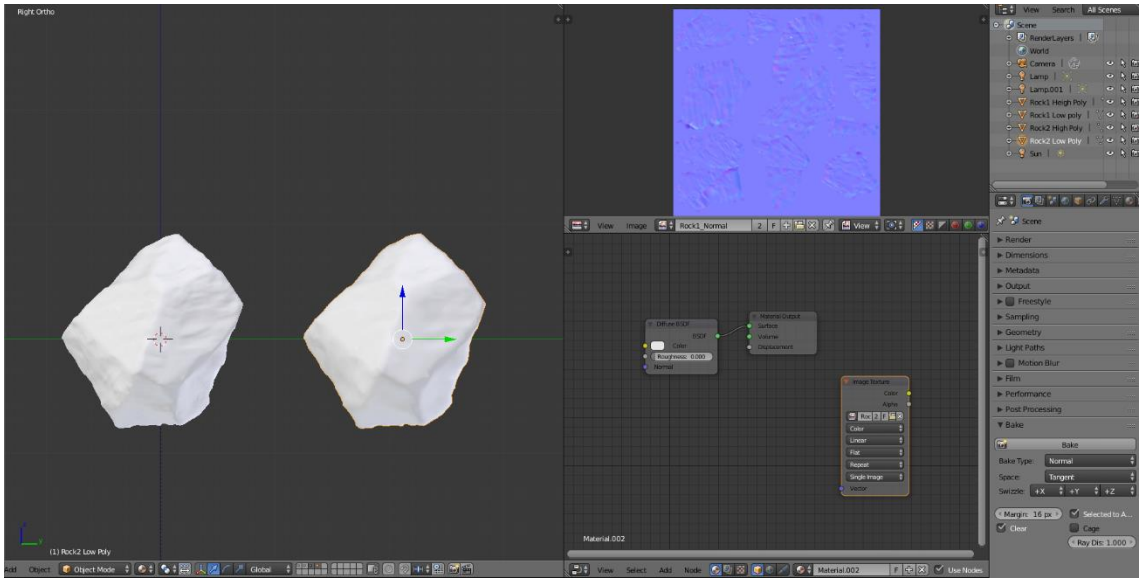

Figura 76: Mapa normal obtenido, y mallas *High Poly* (izquierda) y *Low Poly* (derecha)

Colocando el mapa normal –en el nodo *Image Texture* la opción *Color Space* ha de estar en *Non-Color Data*, este nodo se conecta a un nodo *Normal Map* y éste a su vez en el input *Normal* de *Diffuse BSDF*– en el material de la malla *Low Poly*, se puede ver en *Viewport Shading Material* que su superficie se ve igual a la *High Poly*, como muestra la figura 77.

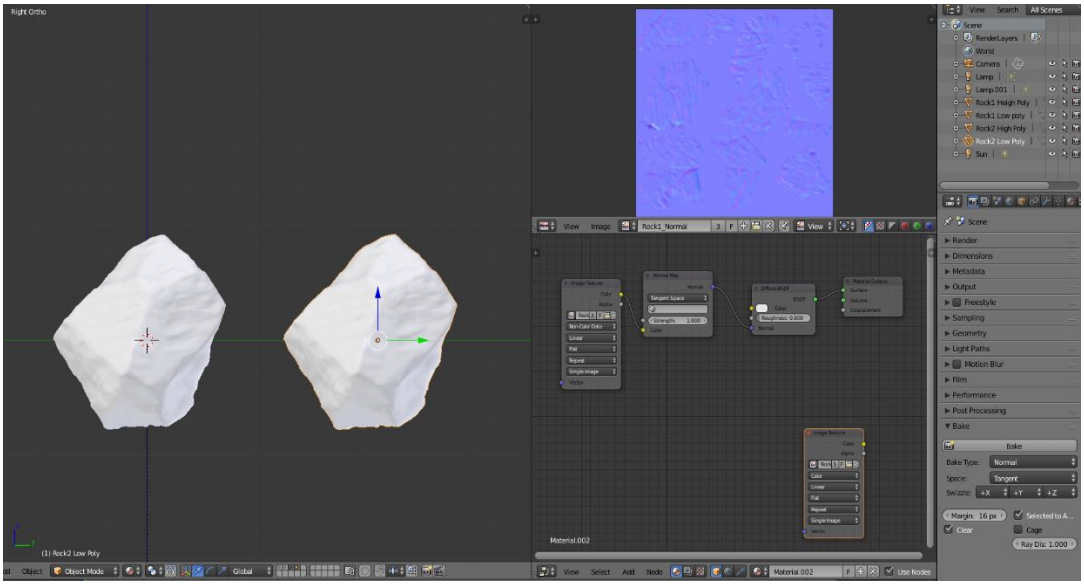

Figura 77: Mallas *High Poly* (izquierda) y *Low Poly* (derecha), teniendo ésta última el mapa normal obtenido con *Bake*

Sin embargo, en Substance Painter se puede hacer esto automáticamente, y permite así texturizar directamente la *Low Poly*.

Se crea en Blender el *UV Map* para esta malla dividiéndola mediante *Mark Seam* y con la opción de *U – Unwrap*, y se exporta como *.fbx*.

En Substance Painter se importa la *Low Poly*, se selecciona la plantilla de *Unreal Engine 4* y la resolución de 4096.

En *Texture Settings – Bake Mesh Maps – Normal* se añade la malla *High Poly* y se hace el *Bake* de las texturas. La malla *Low Poly* ahora con el mapa normal generado parece de nuevo la *High Poly*.

Se empieza con una capa de color base y aumentando *Roughness* para que no tenga brillos. Después, con una capa con información sólo de *Normal* se le añade una

textura de hormigón, la cual viene por defecto, para darle pequeños detalles de rugosidad.

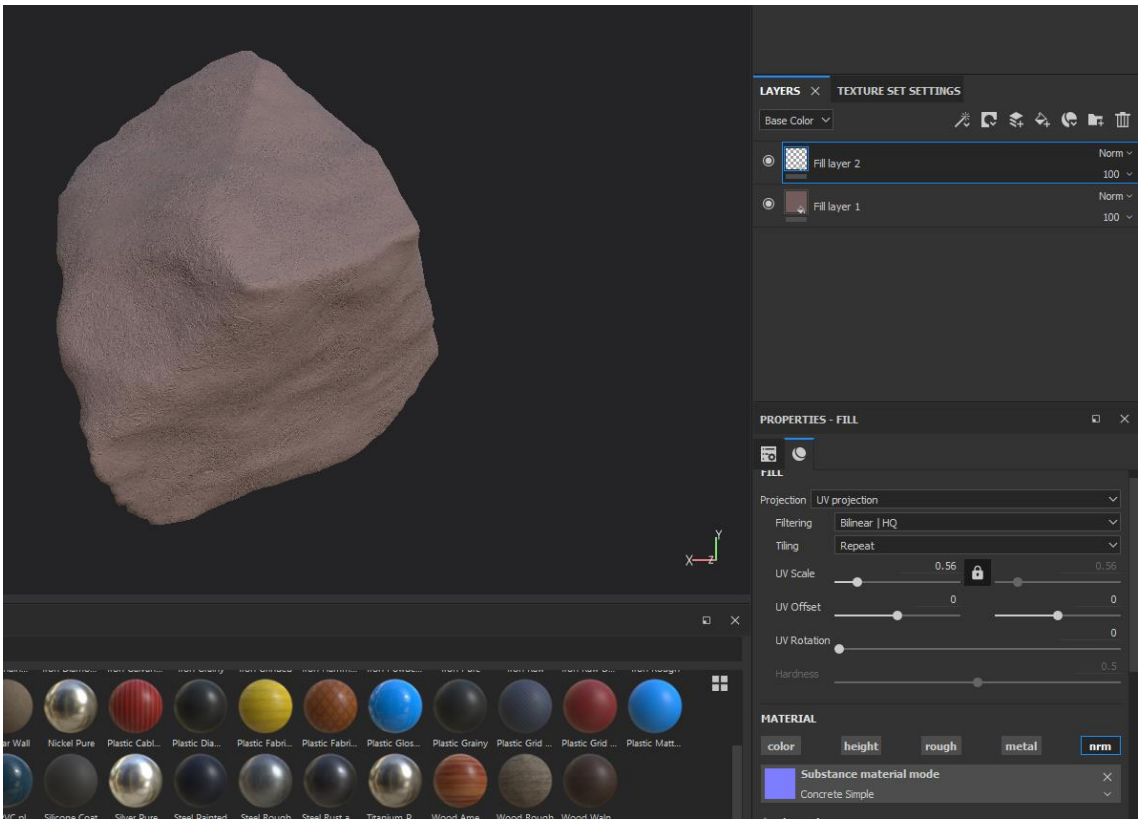

Figura 78: Roca con color base y rugosidad en Substance Painter

Para cada una de las siguientes capas se crea una máscara para que afecten sólo a algunas zonas, que se regulan mediante un editor de máscara. En esta roca, por ejemplo, hay un editor de máscara donde en *Curvature* se selecciona *Cavities*, y en el otro, *Edges*.

En ambos editores de máscara se pone la textura *Grunge Rock*. Se modifican los valores de los diferentes parámetros hasta conseguir el resultado deseado.

Además, a estas capas se les añade *Fill Layer* para tener más variación mediante diferentes texturas *Dirt*, que vienen por defecto. También es posible pintar a mano sobre estas capas con *Add Paint*, escogiendo una brocha *Dirt* y regulando su color en escala de grises, según se quiera aclarar u oscurecer las zonas pintadas.

Figura 79: Roca con máscaras en las esquinas y cavidades; pintando manualmente sobre ellas

Figura 80: Roca texturizada en Substance Painter

5.4.5. Rocas con musgo

Modelado

Al tratarse de rocas pequeñas y lisas, se modelan con pocos polígonos y de la misma forma que las primeras rocas del proyecto, pero sin utilizar el modificador *Displace*. Es decir, a partir de una *Ico Sphere* y deformando ésta en *Sculpt Mode* con las brochas *Sculpt Draw* y *Smooth*. Finalmente, se selecciona el *Smooth Shading*.

Texturizado

Se hace en Blender a partir de nodos con *Cycles Render*. Se combinan dos texturas: una de roca y otra de musgo, mediante un factor que parte de un nodo de ruido (*Noise Texture*). El nodo *ColorRamp* unido a éste (Figura 82) permite alterar la cantidad de área destinada a cada textura: cuanto más blanco haya en la barra, se verá más el musgo, y cuanto más negro haya se verá más roca.

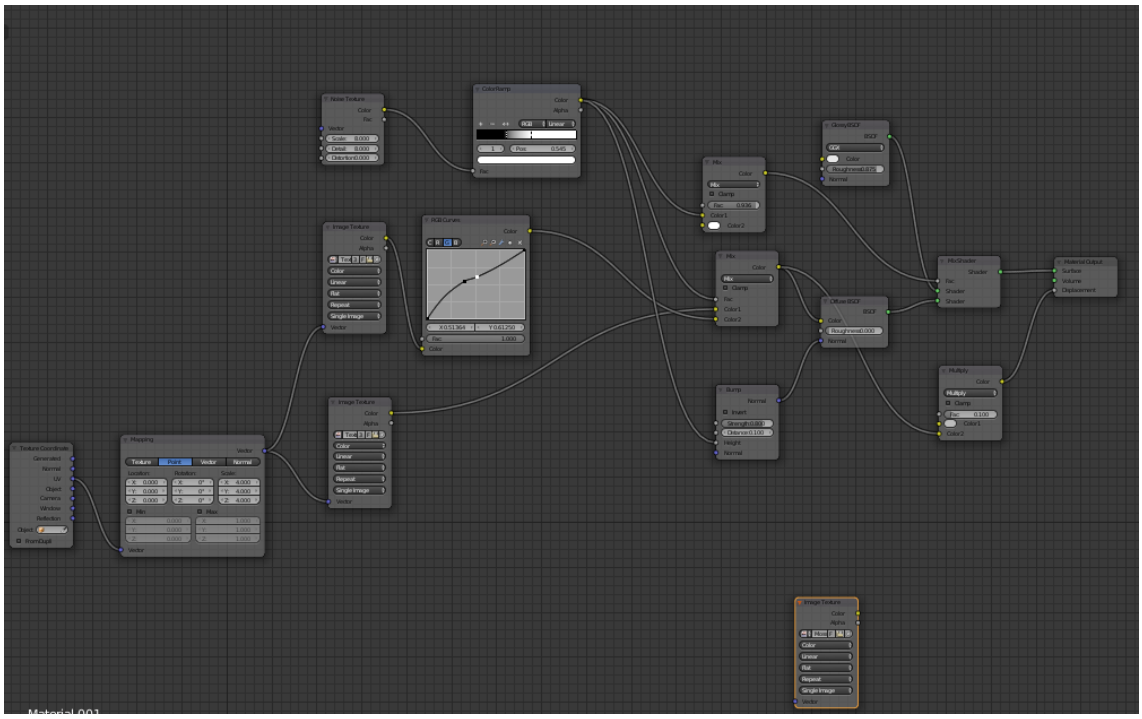

Figura 81: Árbol de nodos del material de las rocas con musgo

El ruido también se utiliza para la entrada *Height* del *Bump*, y añadiendo protuberancias a la superficie, cambiando por tanto el mapa normal.

De este modo con un nodo *Color Mix RGB* se mezclan las dos texturas con el factor determinado por el nodo de *Noise Texture* acoplado al de *ColorRamp*. De aquí sale el color para el *Diffuse Shader* y del *Bump*, la normal.

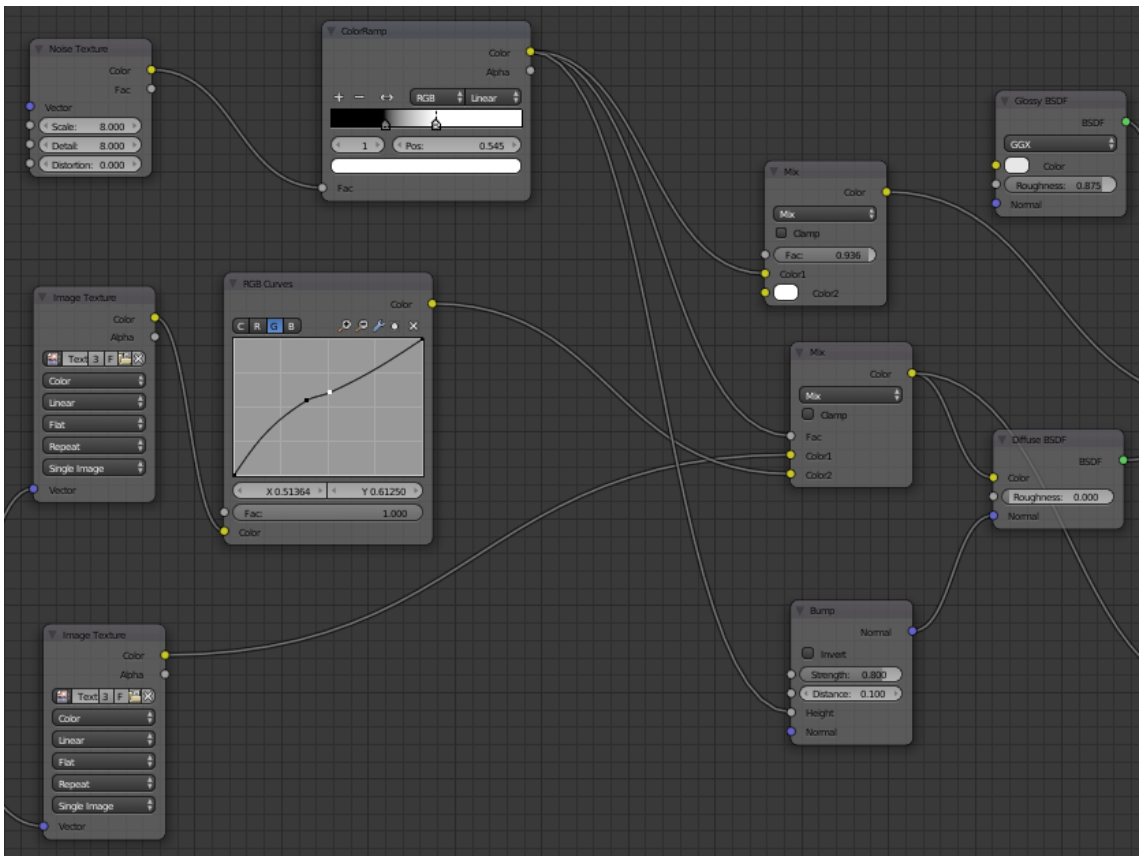

Figura 82: Parte del árbol de nodos de las rocas con musgo, con las dos texturas combinadas

Se modifican las coordenadas de textura mediante los nodos *Texture Coordinate* y *Mapping*. Con el nodo *RGB Curves* varío un poco el color de la textura del musgo antes de combinarla con la textura de roca.

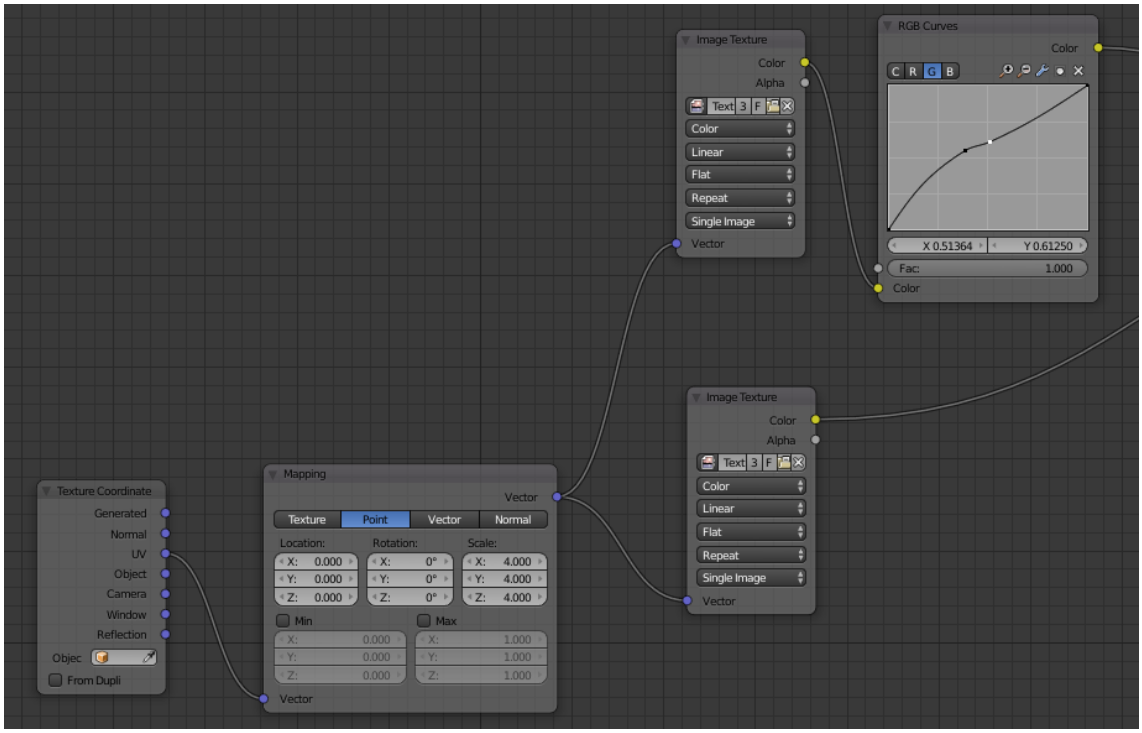

Figura 83: Parte del árbol de nodos de las rocas con musgo, con las dos texturas y sus coordenadas UV

Se utiliza un *Color Mix RGB* a partir del *ColorRamp* del ruido combinado con blanco, para usarlo como factor del *Mix Shader* que combina los shaders *Diffuse* y *Glossy*.

Finalmente, con otro nodo *Color Mix RGB*, escogiendo en *Blend Type* la opción *Multiply*, se le añade el *Displacement*.

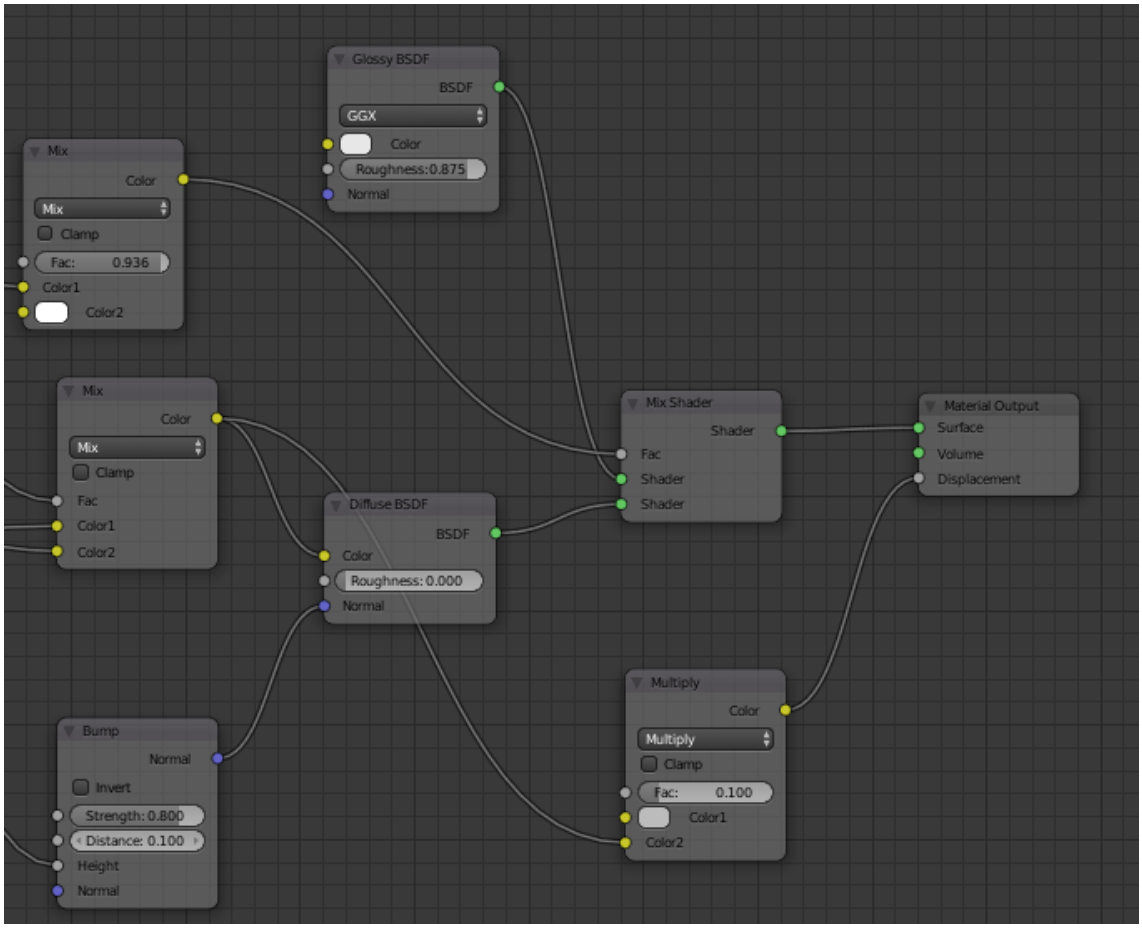

Figura 84: Parte del árbol de nodos de las rocas con musgo con *Material Output*

Figura 85: Roca con musgo vista en el modo *Render* de Blender

5.4.6. Piedras

Modelado

Al tratarse de elementos muy pequeños, tanto su modelado como sus texturas son simples y no es necesario el mapeado. Se modelan como *Ico Sphere* y con las brochas *SculptDraw* y *Smooth* en *Sculpt Mode*.

Texturizado

Se realiza directamente en Unreal con un material sencillo utilizando el color base y un mapa normal. Como color base se usa una textura de roca, y con ella en NormalMap Online se obtiene el mapa normal.

5.4.7. Ramitas

Modelado

Se hacen a partir de un cubo. En *Edit Mode* con éste seleccionado se hace Alt + M – *At Center*, y los vértices quedan todos en el centro. Se añade *Skin Modifier* y *Subdivision Surface* y con la vista *Top Ortho*, mediante la tecla E (*Extrude*) sobre los vértices se van creando las pequeñas ramificaciones, que se pueden ir escalando con Ctrl + A y *Proportional Editing* desactivado. Se marca como raíz la base de la rama seleccionando *Mark Root* en *Skin Modifier*.

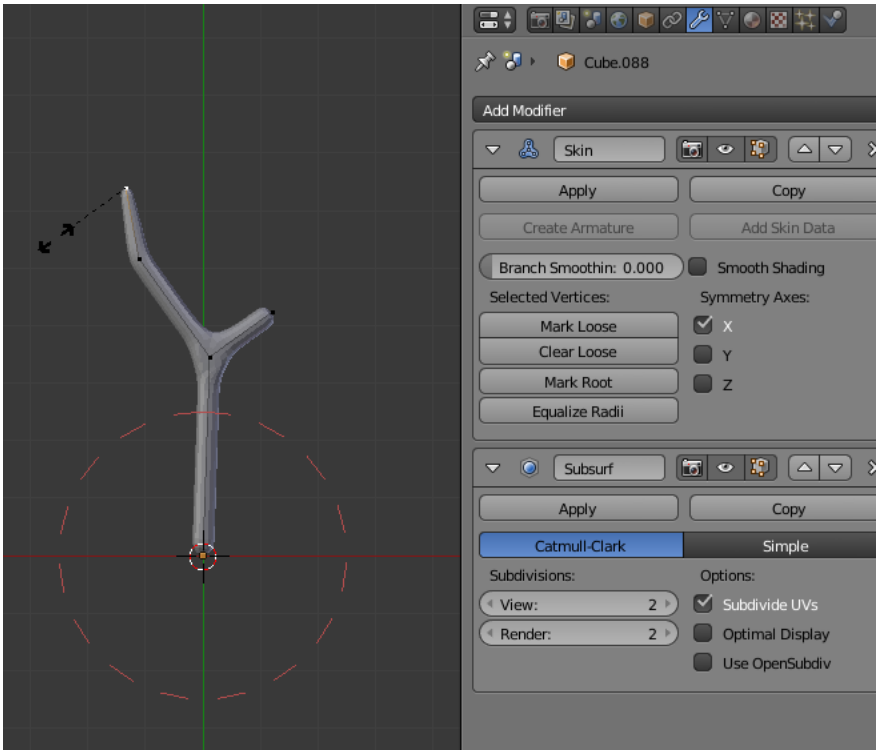

Figura 86: Haciendo ramita utilizando *Skin Modifier*, vista en *Top Ortho*

Tras esto, la ramita tendrá todas las ramificaciones sobre el plano del suelo. Así pues, se desplazan los vértices con *Proportional Editing – Connected* y *Sharp* para que el desplazamiento afecte también a los vértices adyacentes.

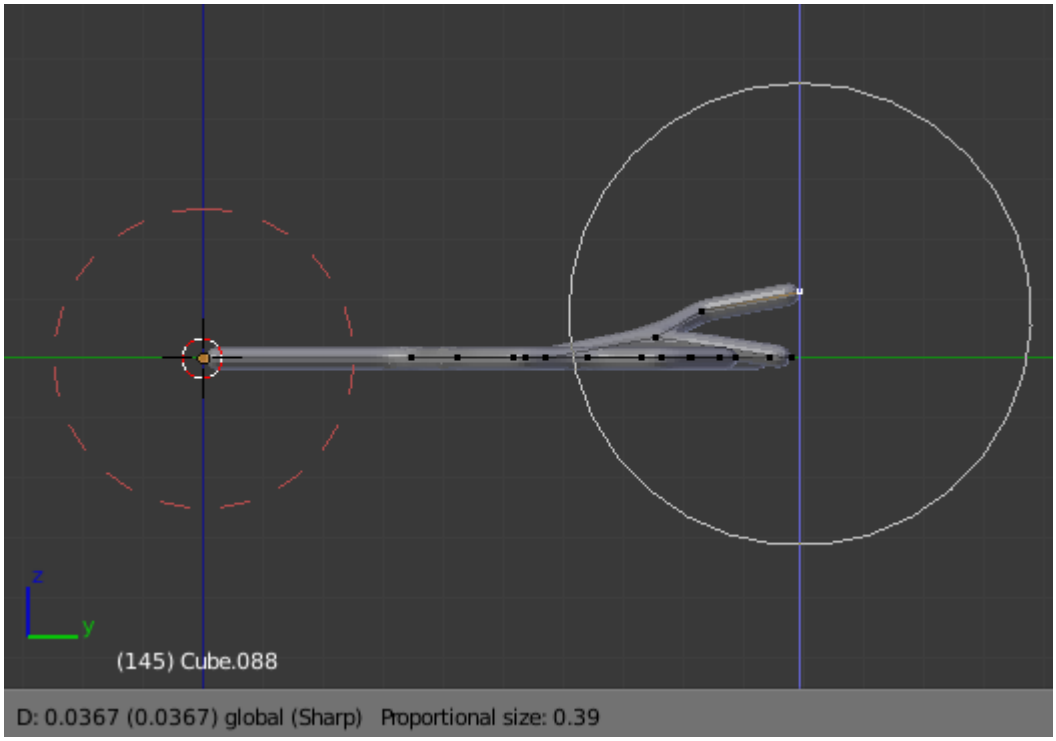

Figura 87: Desplazando vértices de la ramita con *Proportional Editing*, vista en *Right Ortho*

Una vez acabada, se transforma en malla con Alt + C – *Mesh from Curve*. Ahora en *Edit Mode* tiene muchos más vértices, por lo que es necesario reducir esta cantidad: con *Mesh – Clean Up – Limited Dissolve* y con el modificador *Decimate*.

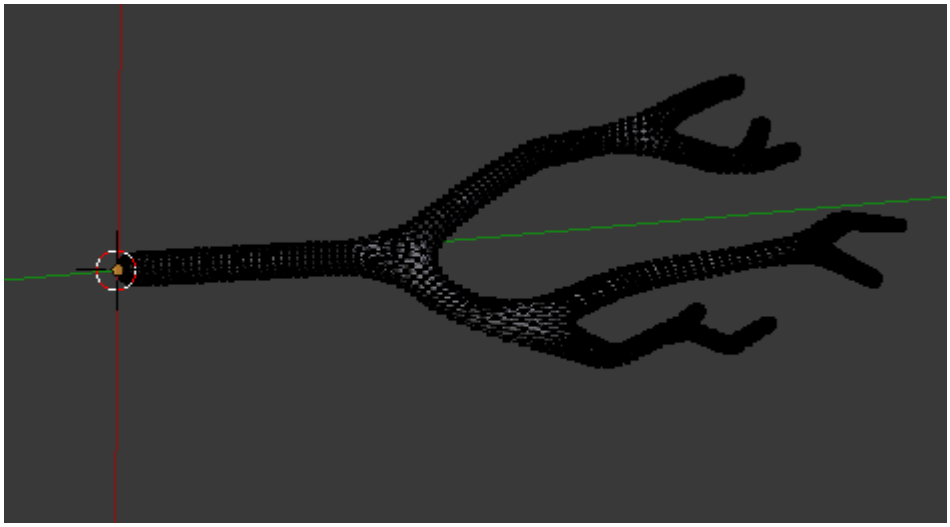

Figura 88: Ramita convertida en malla, vista en *Edit Mode*

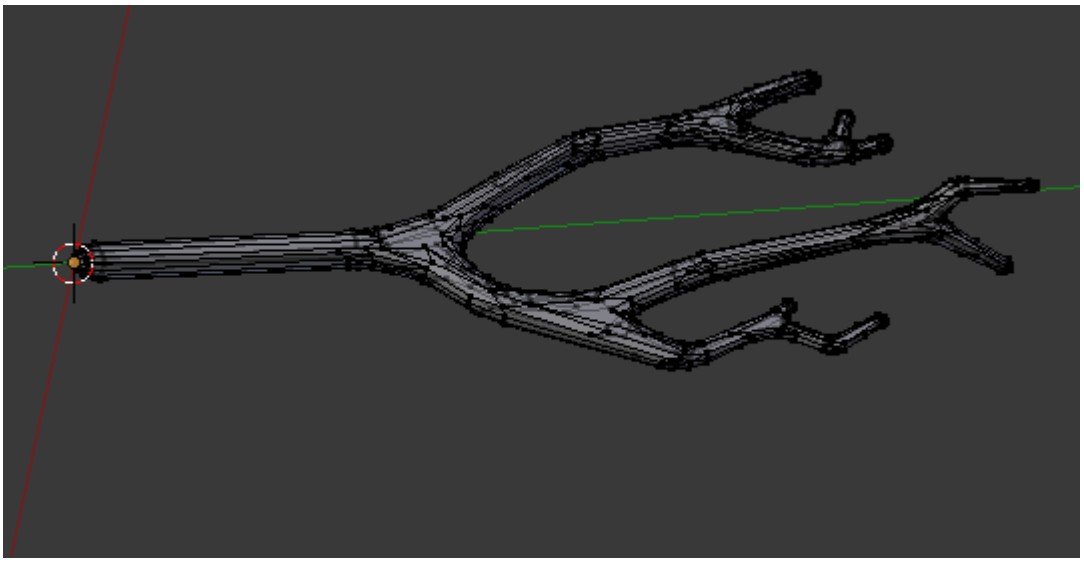

Figura 89: Ramita tras reducir el número de vértices, vista en *Edit Mode*

Texturizado

Se crea el mapa UV con U – *Smart UV Project*. En Blender se obtienen las texturas de color base y normal mediante un material de nodos, utilizando *Bump* a partir de una

textura de ruido modificada con *ColorRamp*, así como desplazamiento. Con *Bake* se generan los mapas de color y normal.

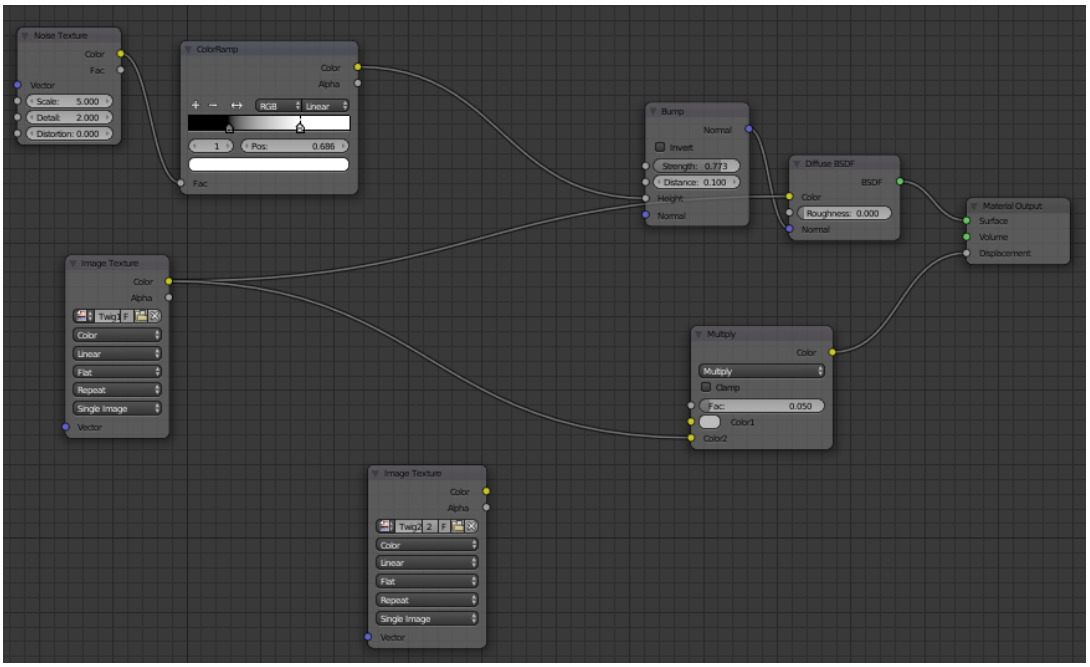

Figura 90: Árbol de nodos del material de las ramitas

Figura 91: Ramita vista en *Rendered Viewport*

5.4.8. Árboles

Modelado

Para los árboles se activa el addon *Add Curve: Sapling Tree Gen* en *User Preferences*. Así, con Shift + A – *Curve* – *Sapling Tree Gen* aparece un menú con diferentes parámetros para crear árboles, así como arbustos o pequeñas plantas. Por defecto se crea una curva con forma de ramas de árbol.

Se hace uso especialmente cuatro submenús del addon:

- *Geometry*: se activa *Bevel* para darle volumen a la curva. Aquí se edita la resolución, la escala y distribución de las ramas.
- *Branch Splitting*: se editan la cantidad de ramas, los niveles (ramas saliendo de ramas), la división de una en dos o más ramas, etc.

• *Branch Growth*: para la rotación y escalado de ramas, así como su curvatura.

• *Leaves*: se añaden las hojas, escogiendo la cantidad, la forma (se coge la de rectángulo), distribución, escala y rotación.

En el mismo addon hay además un submenú de *Armature* y otro de *Animation*. Al principio, se utilizaron éstos para darle movimiento al árbol, simulando viento. En *Animation* se editan la cantidad de viento, así como la amplitud y frecuencia del movimiento. El resultado parecía demasiado artificial si sólo se quiere poner un viento leve, ya que movía las ramas, y las hojas no tenían movimiento independiente a éstas. Así pues, se decide añadir el efecto de viento en Unreal.

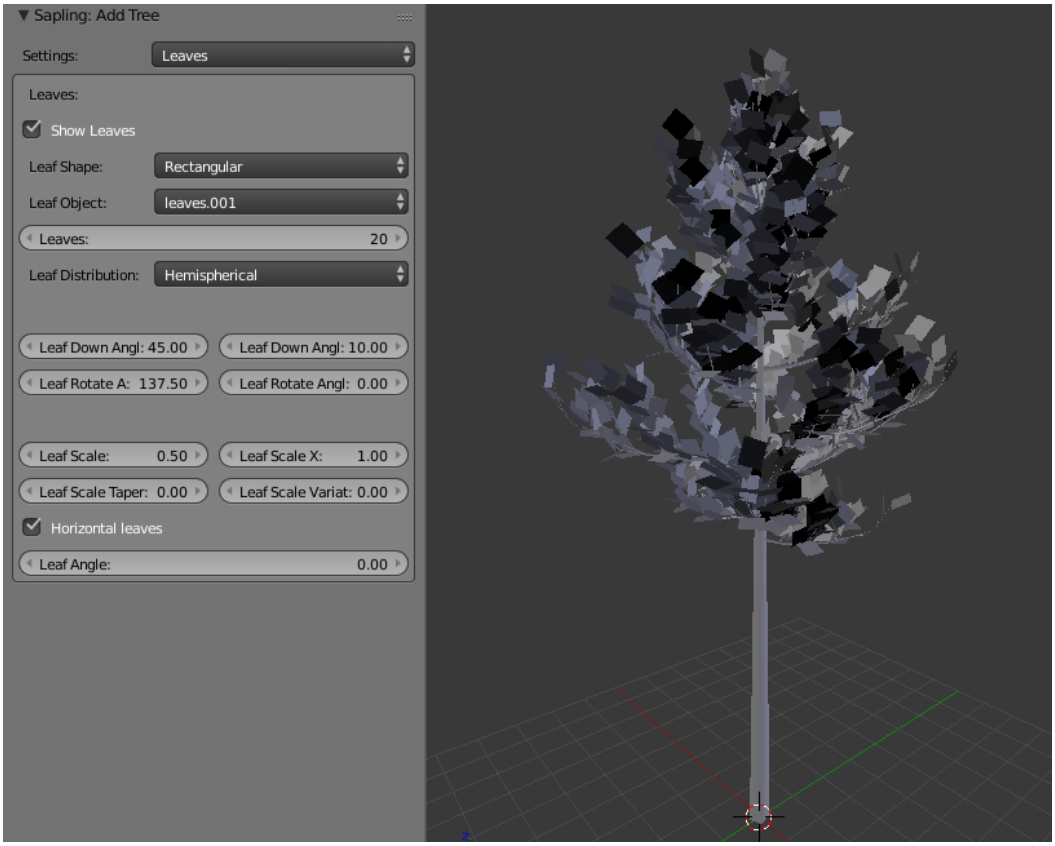

Figura 92: Creando un árbol con el addon *Sapling Tree Generator*

Tras acabar el modelado con el addon, habrá dos objetos: uno con el tronco y las ramas, y otro con todas las hojas.

Se convierte el tronco, que es una curva, en una malla con Alt + C – *Convert to Mesh from Curve* en *Object Mode*.

Finalmente, para que el árbol se vea más natural, se añade variación en el escalado de las hojas, de modo que no tengan todas el mismo tamaño. Se selecciona el objeto con las hojas y en *Edit Mode*, con *Select – Random* y el modo de selección *Faces*, se seleccionan algunas de las hojas. El punto de pivote ha de estar en *Individuals Origins*, para que no queden las hojas separadas de sus respectivas ramas al escalar. Se hacen más pequeñas o más grandes, se deseleccionan y volviendo a hacer *Select – Random* se escala otro grupo de hojas.

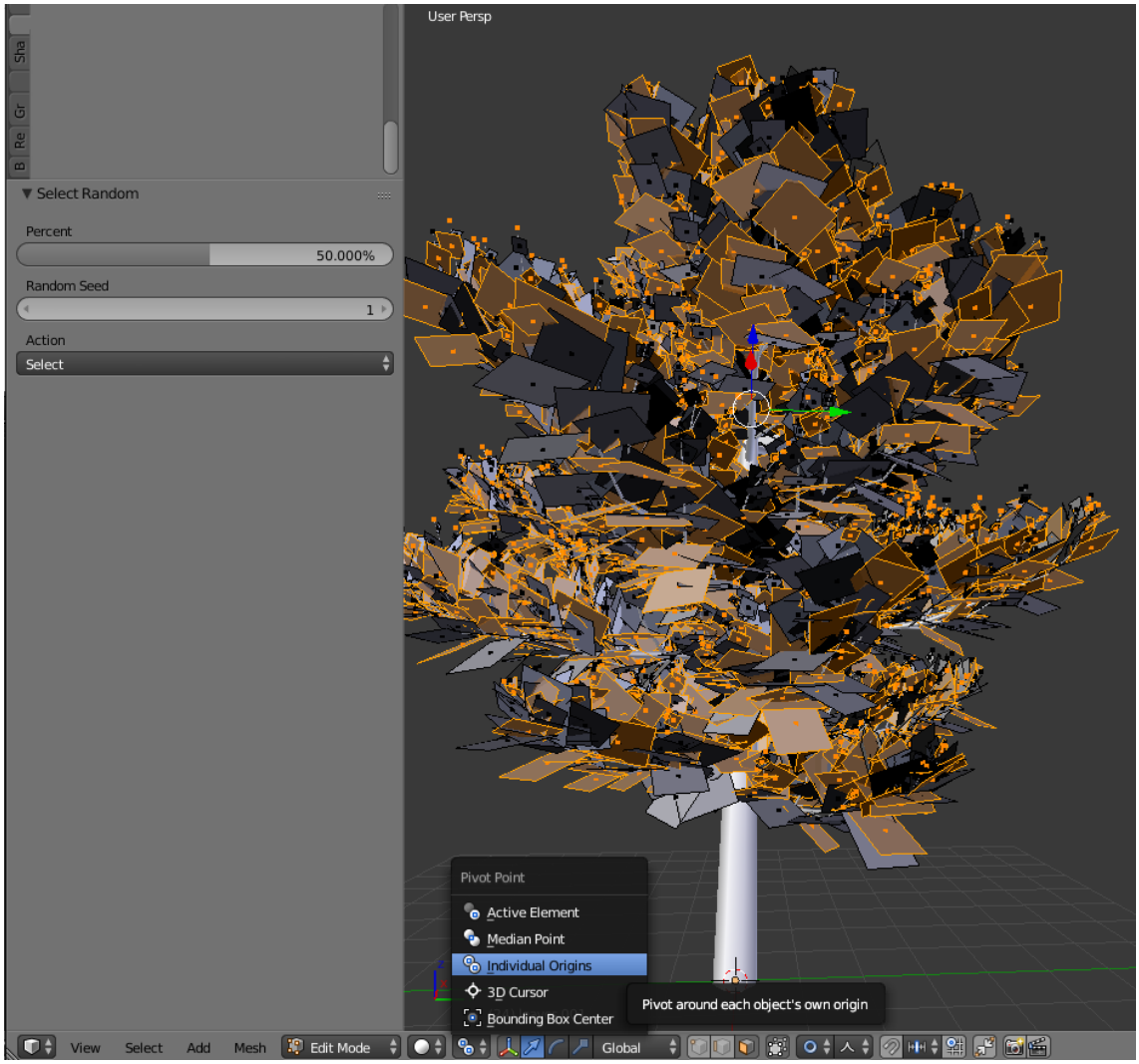

Figura 93: Selección aleatoria de hojas previamente al escalado

Texturizado

Para el tronco y las ramas del árbol, se utiliza *Smart UV Project*. Primero, seleccionando todo el objeto, y después seleccionando únicamente las caras pertenecientes al tronco, puesto al ser el tamaño de éstas diferente al de las caras en las ramas, la textura se vería también de manera diferente (estaría borrosa en el tronco).

El material en Blender es simple:

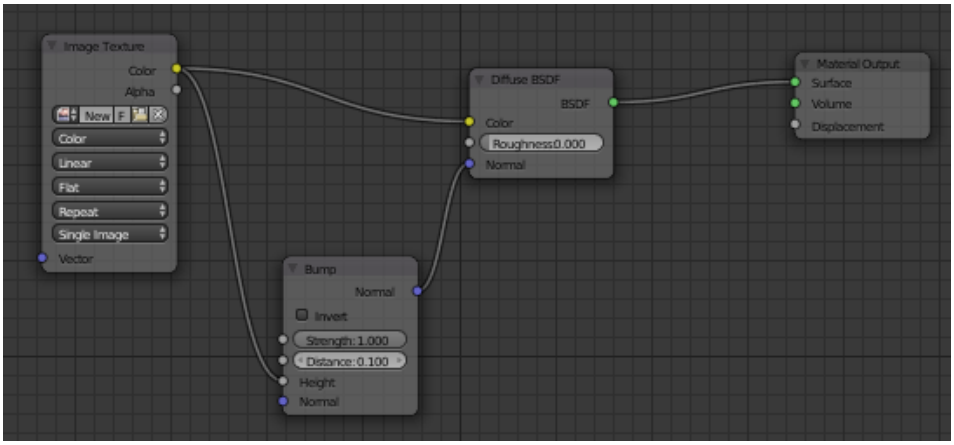

Figura 94: Material en Blender del tronco y las ramas del árbol

Se obtiene el mapa de *albedo* en Blender, y a partir de éste, los de *ambient occlusion* y *normal* en NormalMap Online. Se utilizan como materiales las instancias del material *TreeBark_Master_Mat*.

Para las hojas se añaden los nodos de *Bump*, *Glossy BSDF*, *Transparent BSDF*¸ y *Translucent BSD.*

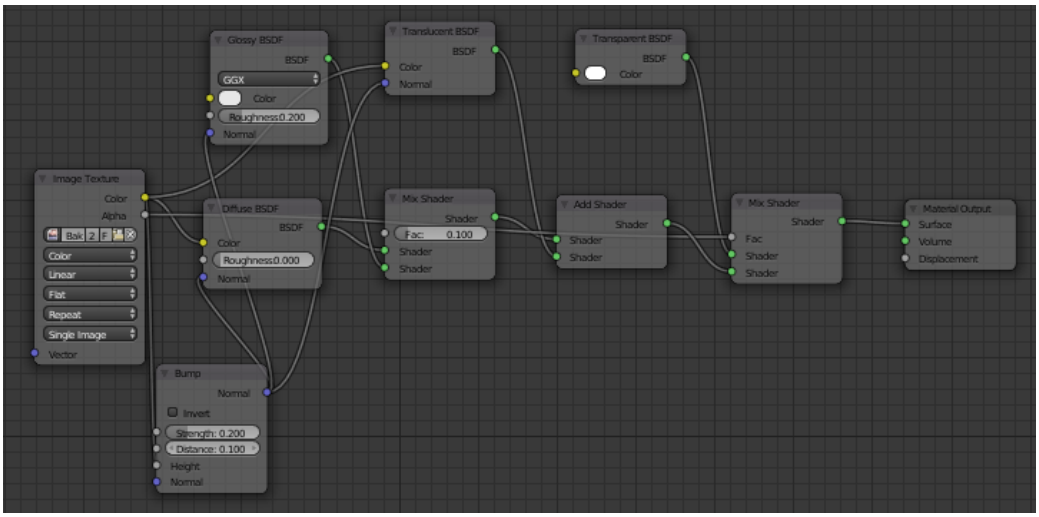

Figura 95: Material en Blender de las hojas del árbol

Figura 96: Árbol en Blender, visto en *Rendered Viewport*

Figura 97: Árbol en Unreal Engine 4

5.4.9. Flores

Modelado

El mismo addon utilizado para los árboles se usa para hacer el tallo y los pedúnculos de las flores. En este caso el tamaño es menor y no se añaden ramas, pero se divide la única que hay en "ramas" separadas (pedicelos). Tampoco se ponen las hojas.

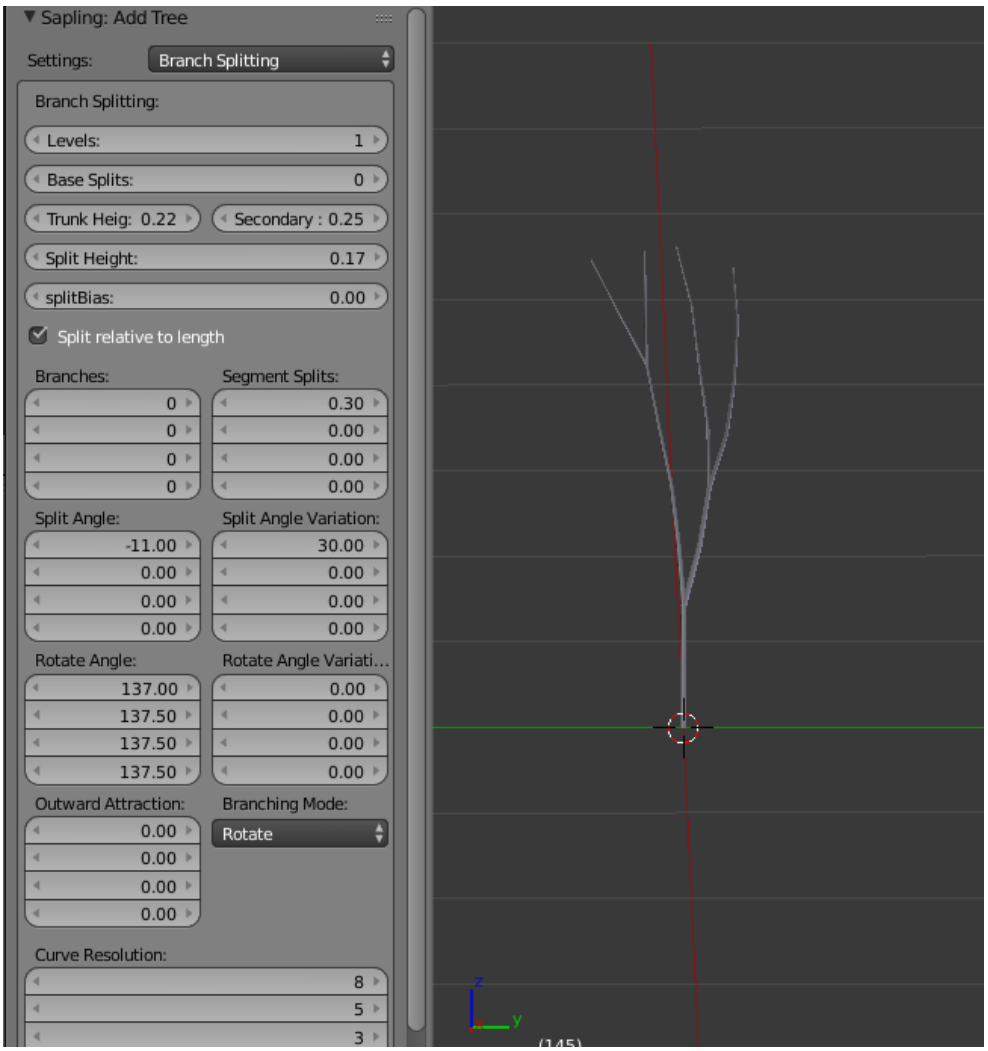

Figura 98: Creando flor con el addon *Sapling Tree Generator*

Como habrá más flores y todas tienen una forma similar, antes de quitar el menú de addon se selecciona *Export Preset* y se pone un nombre. Al volver a utilizar el addon para crear una flor nueva, se comenzará con este *Preset* y se varían los parámetros a partir de ahí.

Al igual que con los árboles, se transforma la curva en una malla.

Para poner las flores y las hojas se utilizan sistemas de partículas.

Primero, en *Properties – Data* se crean dos *Vertex Groups*: en uno se selecciona con la tecla C dónde deben ir las hojas, y en el otro, dónde deben ir las flores.

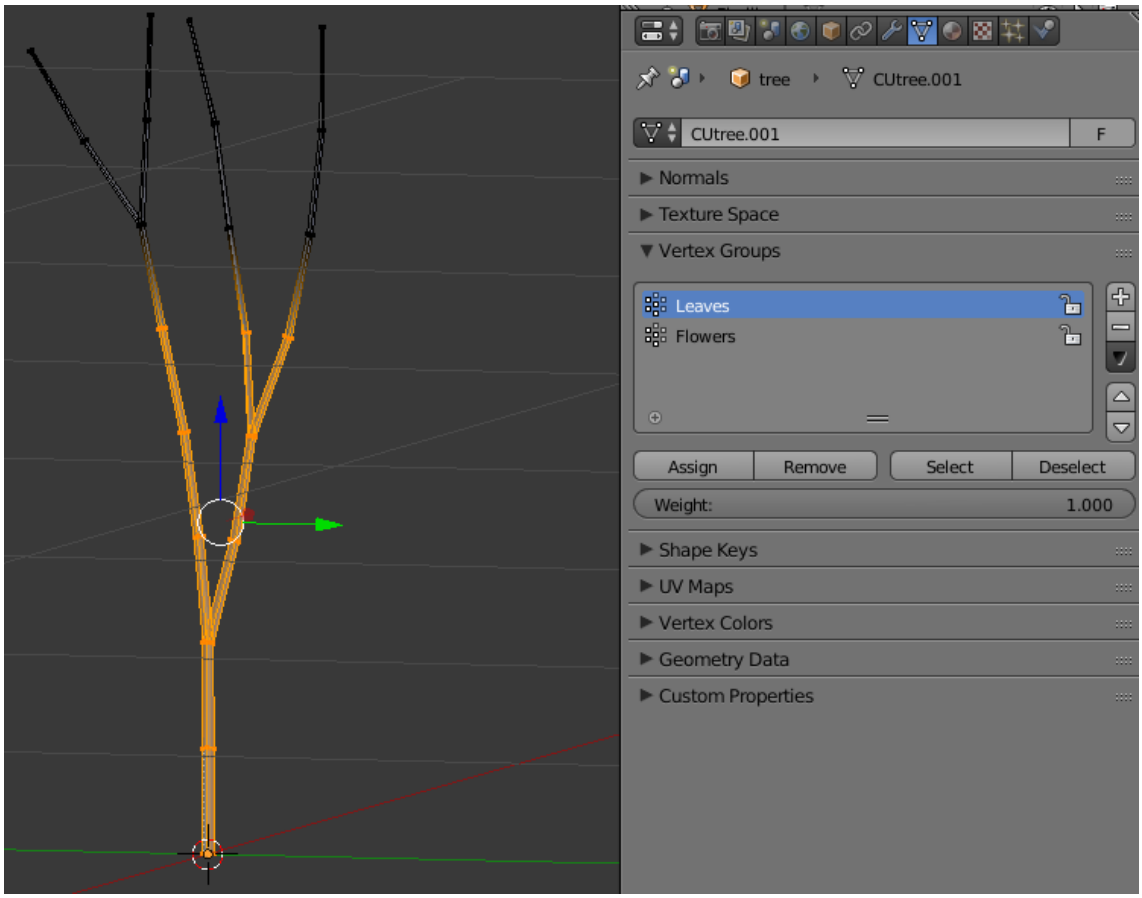

Figura 99: Seleccionando dónde deberían ir las hojas con *Vertex Groups*

Se importa con *Images as Planes* las imágenes de la flor y la hoja a colocar. Se ponen en la posición y rotación deseadas. Con Ctrl + R se añaden vértices para desplazarlos y que así la superficie no sea completamente plana. Para el caso de las hojas, se cambia el punto de pivote con Ctrl + Alt + Shift + C para ponerlo en el extremo del pecíolo.

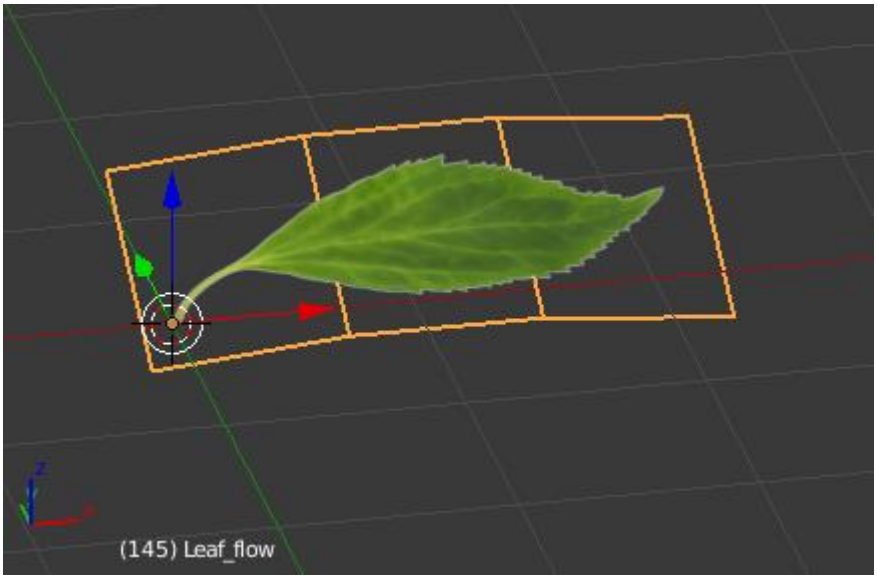

Figura 100: Hoja a utilizar en el sistema de partículas

Tanto para la flor como para la hoja, se crean sistemas de partículas de tipo *Hair* y *Advanced*. Se escoge una cantidad de partículas pequeña. En el apartado *Render* se selecciona el objeto a usar como partícula (el plano de la flor o la hoja) y *Rotation* para que utilice la rotación que se le puso a este objeto.

Después, en *Vertex Groups – Density*, se introduce el *Vertex Group* creado anteriormente para que las partículas sólo aparezcan sobre la parte que deben.

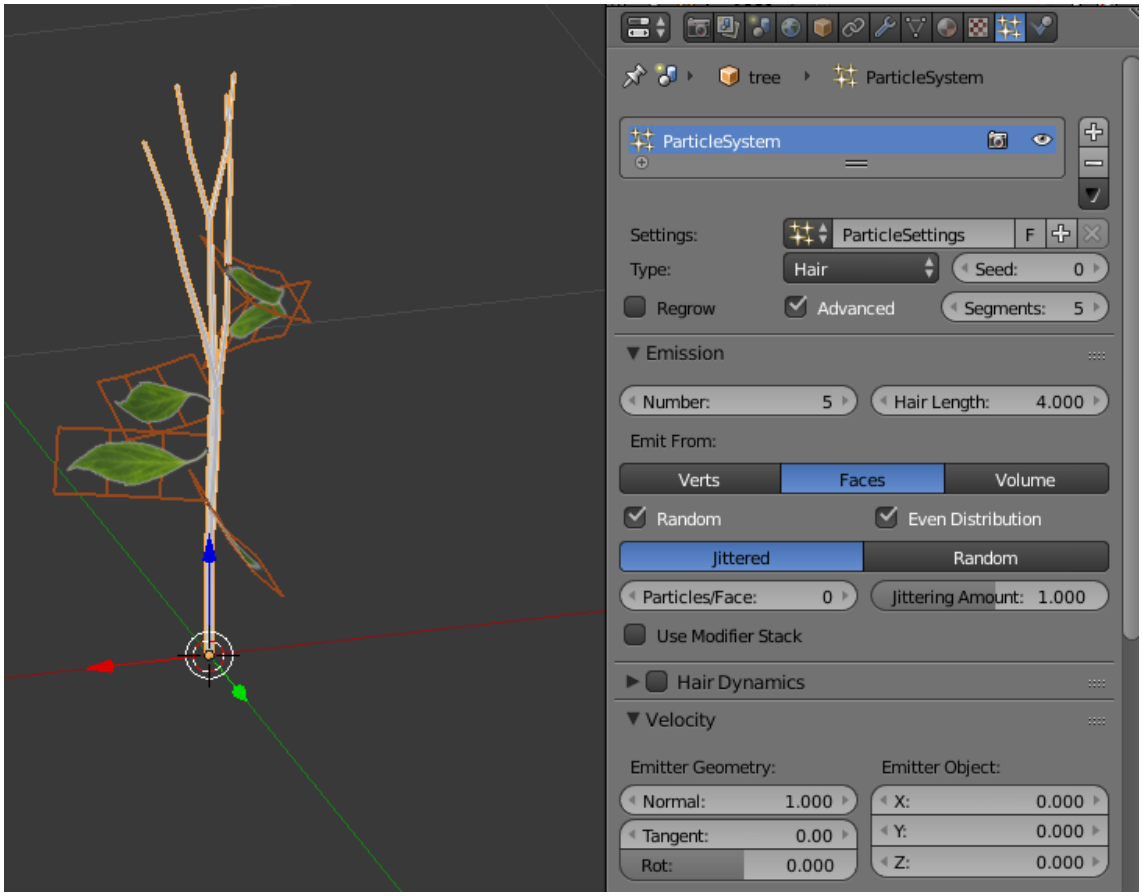

Figura 101: Sistema de partículas de hojas para las flores (1)

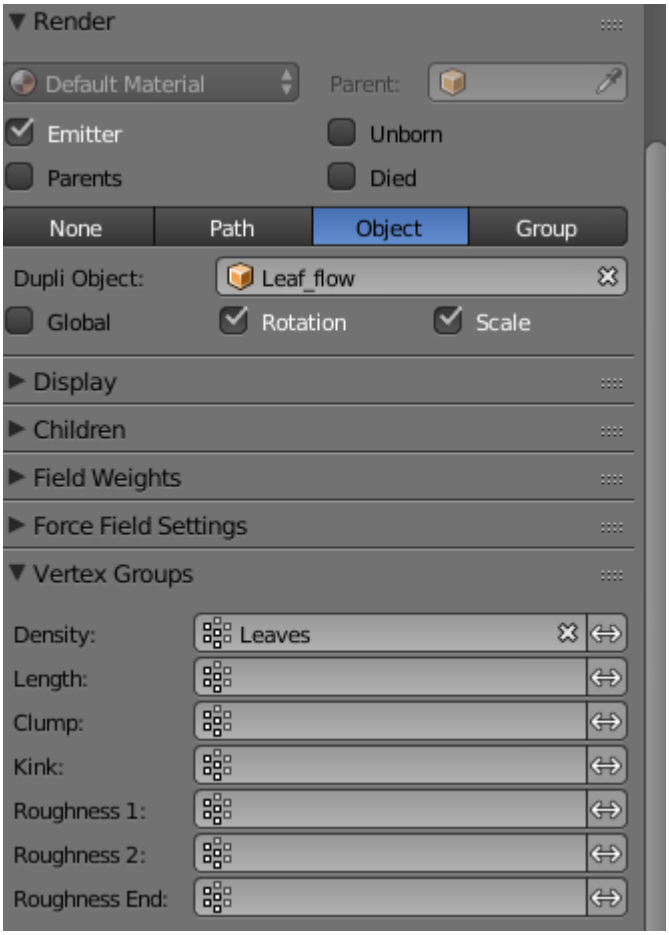

Figura 102: Sistema de partículas de hojas para las flores (2)

Texturizado

Los mapas de textura de las hojas y las flores se obtienen mediante nodos en Blender, de igual manera que para las hojas caídas. El mapa normal se genera con NormalMap Online, y el de opacidad con Photoshop. En Unreal se utilizan instancias del material *Foliage_Master_Mat* con los respectivos mapas de textura. A ambos se les añade algo de viento.

En cuanto al tronco, la textura consiste en un sólo color, ya que no se necesita más detalle al ser tan pequeño y fino. El material en Unreal consiste en el nodo principal y el nodo con el color.

Figura 103: Flor vista en *Rendered Viewport*

5.4.10. Hierbas y plantas

Modelado

Se importa la imagen con *Images as Planes* y se coloca con la posición y rotación deseadas. Con Ctrl + R se añaden vértices para curvarlas.

En el caso de la hierba, con el punto de pivote en *3D Cursor* y éste en el centro de coordenadas, se duplica el plano las veces que se quiera con Shift + D y se rota con R en el eje Z (se hace por defecto si se rota con la vista en *Top Ortho*).

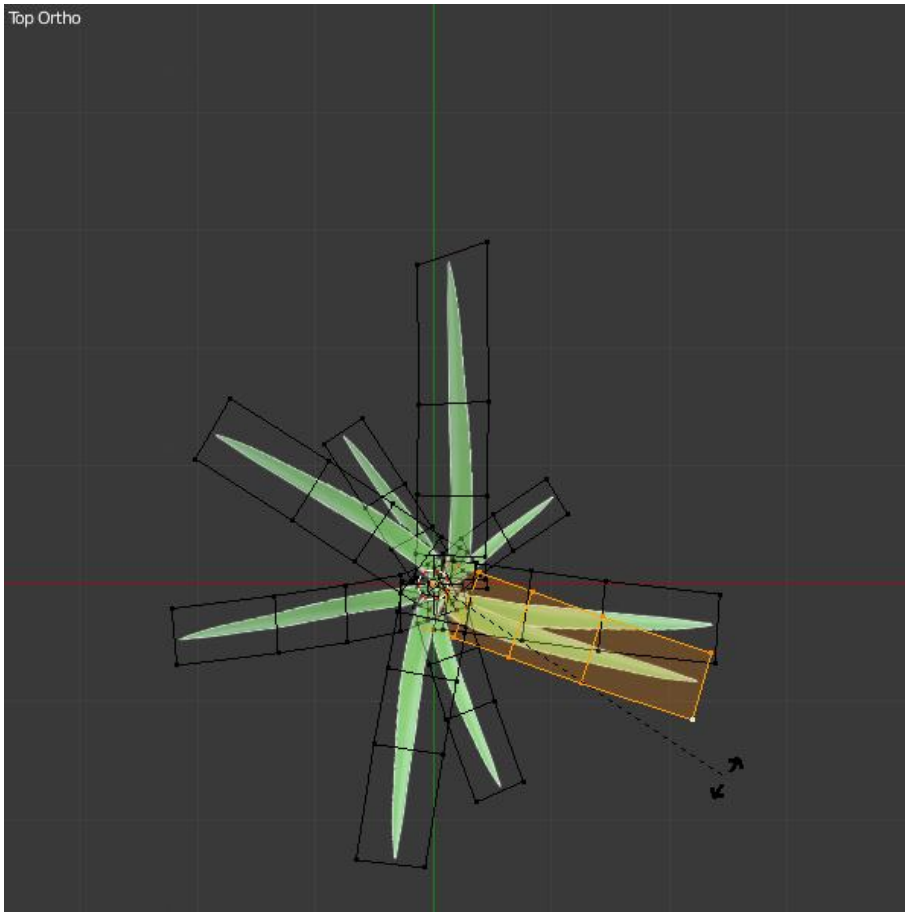

Figura 104: Haciendo un grupo de hojas de hierba en *Edit Mode*

Seleccionando cualquier vértice, con la tecla L se puede seleccionar todo el plano, para mover o escalar cada uno por separado. Además, con *Proportional Editing* en *Connected* y *Sharp* se pueden mover los vértices afectando a los cercanos, para que se muevan a su vez.

Figura 105: Desplazando vértices con *Proportional Editing* en *Edit Mode*

Por otro lado, para los otros tipos de planta se utilizan diferentes imágenes (importadas con *Images as Planes*), también deformando los planos.

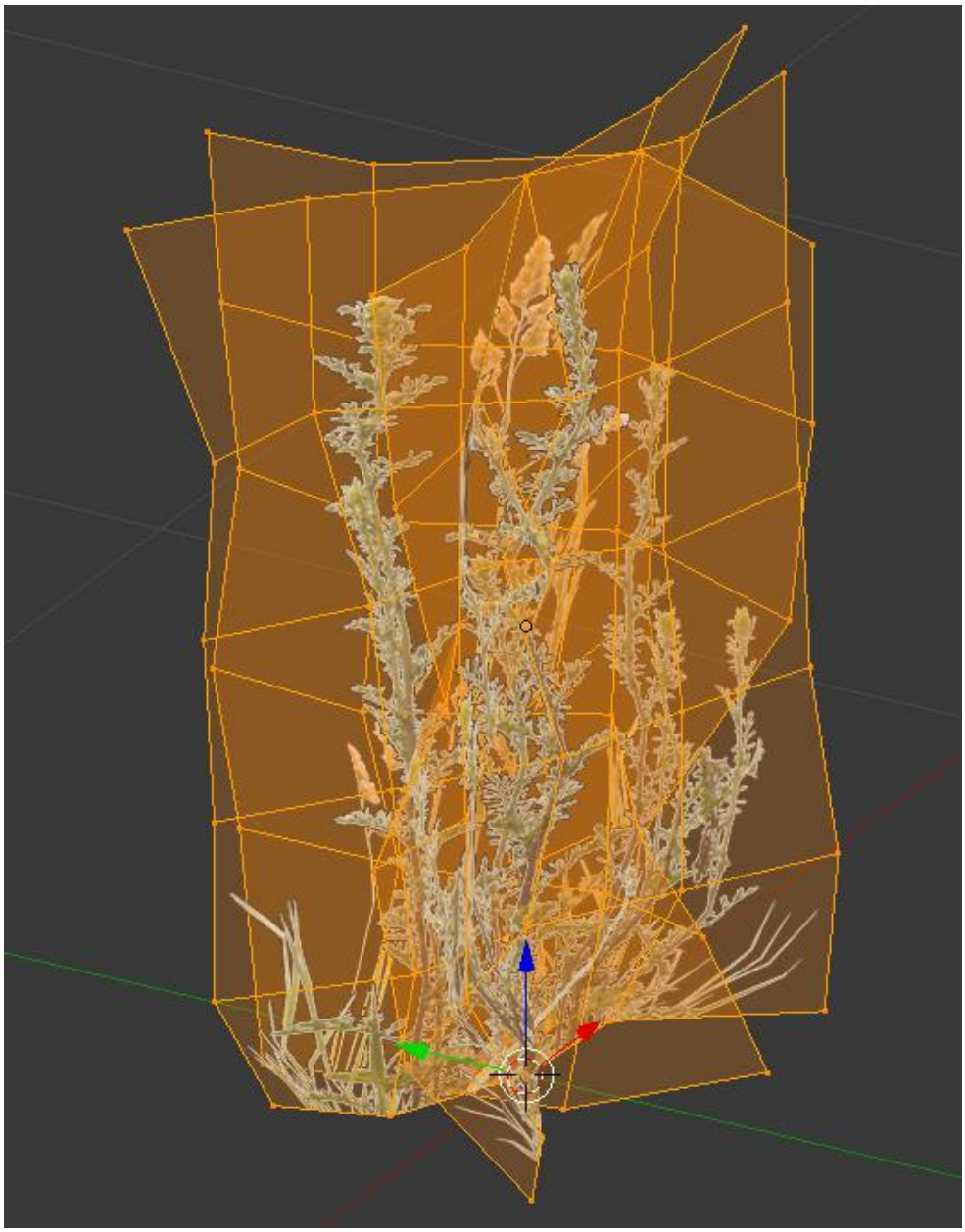

Figura 106: Planta compuesta por varios planos deformados, vista en *Edit Mode*

De nuevo, se crean los mapas necesarios para el *Master Material* de Unreal mediante Blender, NormalMap Online y Photoshop.

Figura 107: Hierbas en Blender, vistas en *Rendered Viewport*

Figura 108: Planta en Blender, vista en *Rendered Viewport*

Los niveles de detalle de estos objetos son realizados en Blender, eliminando vértices manualmente. Después se importan en Unreal. Esto fue debido a que los que se crean en el motor automáticamente eliminaban partes de la malla que debían seguir siendo visibles a distancia.

5.4.11. Pájaro

Primer pájaro

En un principio se hizo toda la animación del pájaro –tanto el movimiento de las alas como el desplazamiento– en Blender.

Modelado

Para el cuerpo, se parte de un plano del cual se corta la mitad (teniendo la vista en *Top Ortho*) y al que se le añade el modificador *Mirror* sobre el eje x. De este modo lo que se realice en el lado "real" del objeto se hará de igual forma sobre el lado "espejo", así que el pájaro será simétrico en el eje x. Se crean nuevas caras con Ctrl + R y moviendo los vértices se crea la forma vista desde arriba.

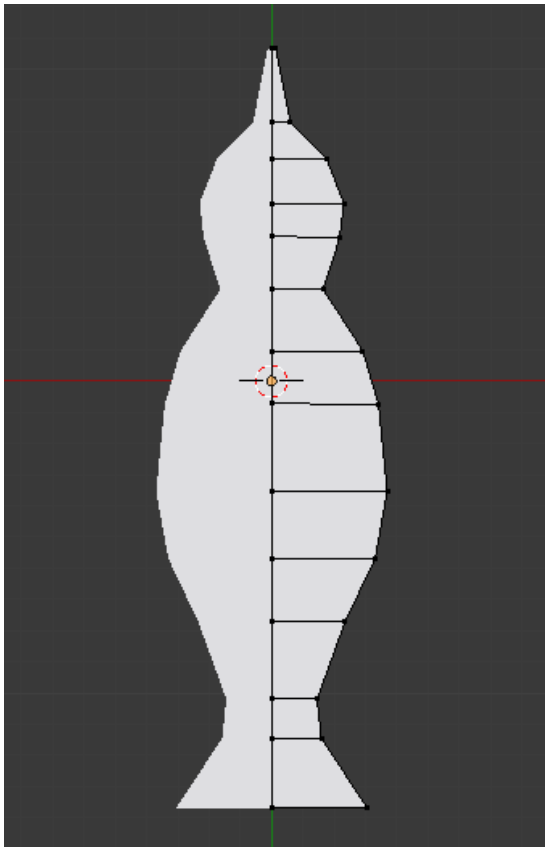

Figura 109: Cuerpo del pájaro en un plano, utilizando *Mirror Mode*, visto en *Edit Mode* con *Top Ortho*

Una vez hecha, con *Extrude* se añade volumen en el eje *z* y añadiendo y desplazando vértices nuevos se completa el modelado del cuerpo. Después se le aplica el modificador *Mirror* y *Smooth Shading*.

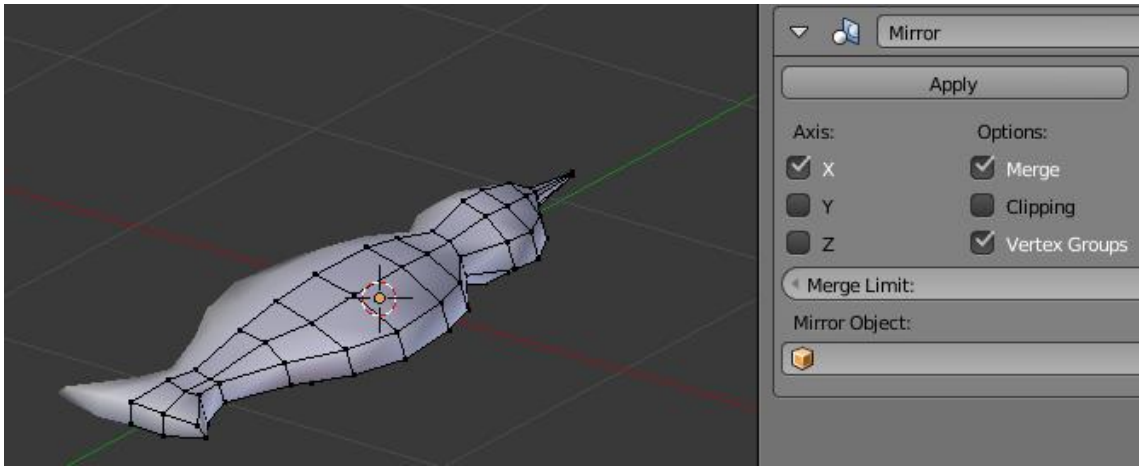

Figura 110: Cuerpo del pájaro con volumen, utilizando *Mirror Mode*

En cuanto a las alas, se utiliza *Import Images as Planes* y se colocan a los costados del cuerpo creado. Seleccionando éstas y el cuerpo, con Ctrl + J se une todo como una única malla.

Texturizado

Hasta comprobar el funcionamiento en Unreal, se mantienen las texturas lo más simples posibles. Al cuerpo se le pone un color, y las alas ya tenían como textura la imagen con la que se habían importado.

Animación

En primer lugar, se crea el esqueleto del pájaro. Con la vista de nuevo en *Top Ortho* y haciendo Shift + A – *Armature* – *Single Bone*, se añade la armadura en el cuerpo. Luego en *Edit Mode* haciendo *Extrude* sobre la esfera inicial se añade también en cada ala. En los

datos del objeto *armature*, en el subapartado *Display*, se selecciona *X-Ray* para que no quede oculto dentro de la malla del cuerpo.

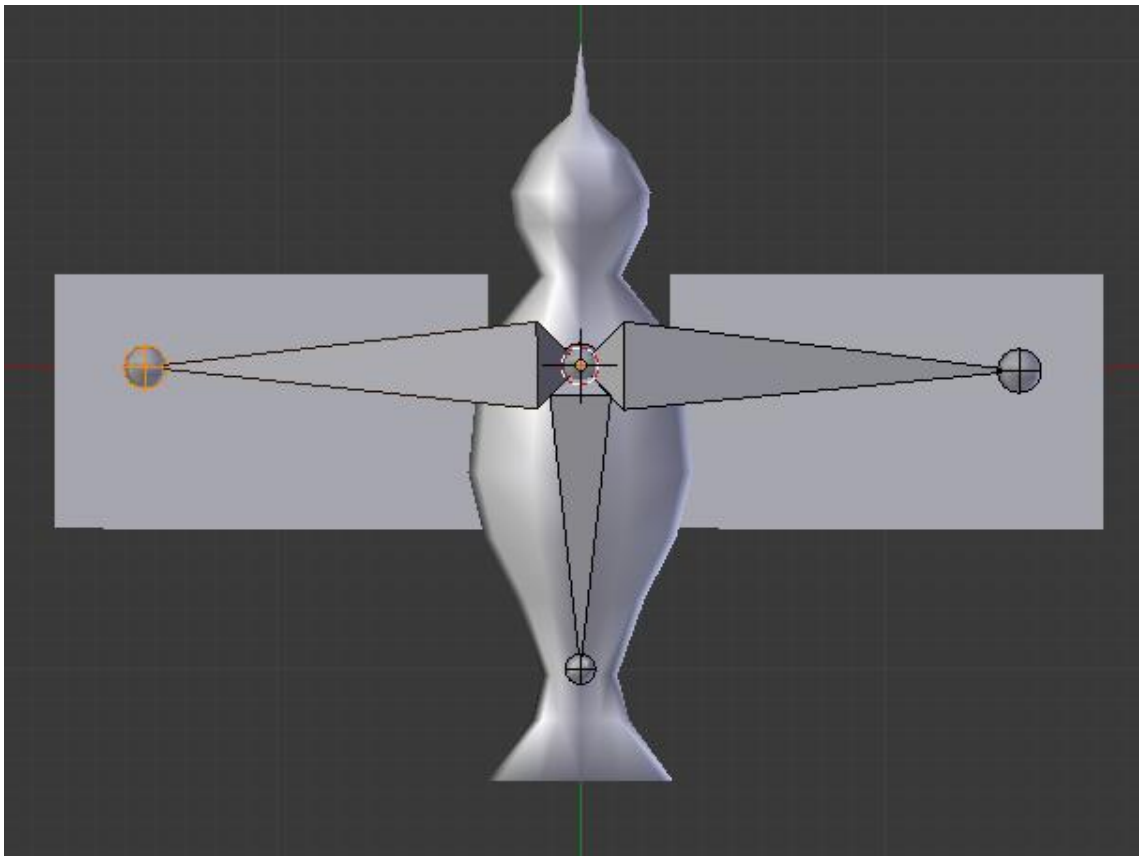

Figura 111: Haciendo la armadura del pájaro, visto en *Edit Mode* con *Top Ortho*

Se ha de hacer que la armadura sea padre de la malla del pájaro: se selecciona primero la malla (hijo) y luego a la armadura (padre), Ctrl + P y *Armature deform with automatic weights*. Con la armadura seleccionada, en *Pose Mode* se rota el "hueso" de alguna de las alas, y el ala de la malla se mueve con él.

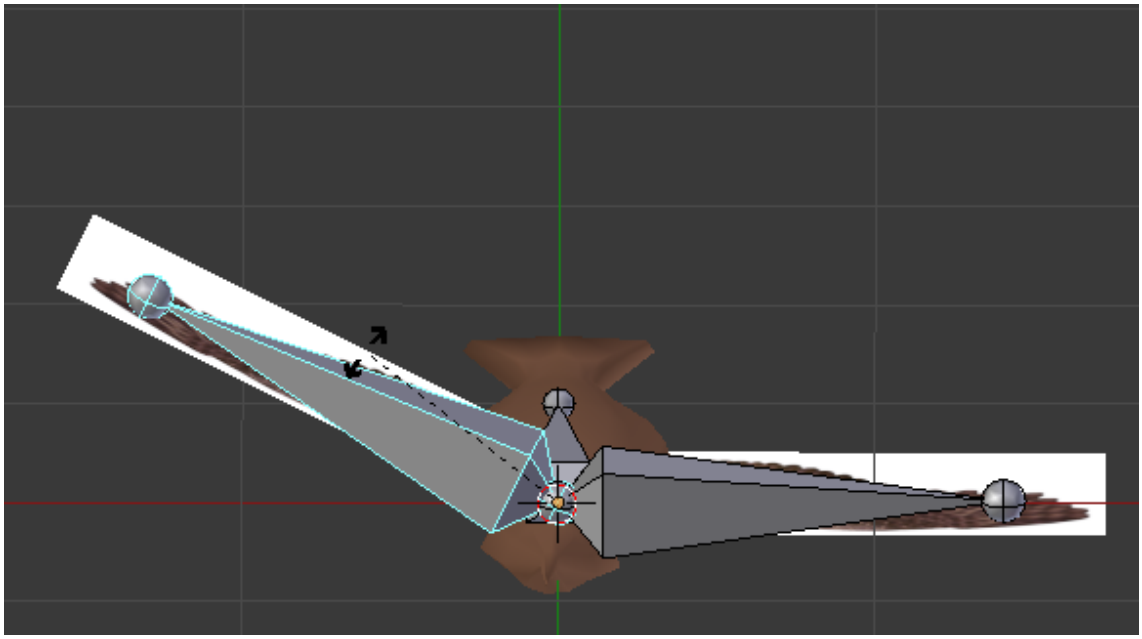

Figura 112: Moviendo *armature* de un ala con la malla siguiendo el movimiento, en *Pose Mode*

Como se ve en la figura 112, el desplazamiento no sólo afecta en la malla al ala sino también a parte de la cabeza, que se deforma. Para solucionar esto, con la armadura seleccionada en *Pose Mode*, se hace click sobre la malla del pájaro y en el modo *Weight Paint*, para cada "hueso" de ala, se pinta la parte de la malla a la que debe afectar su movimiento.

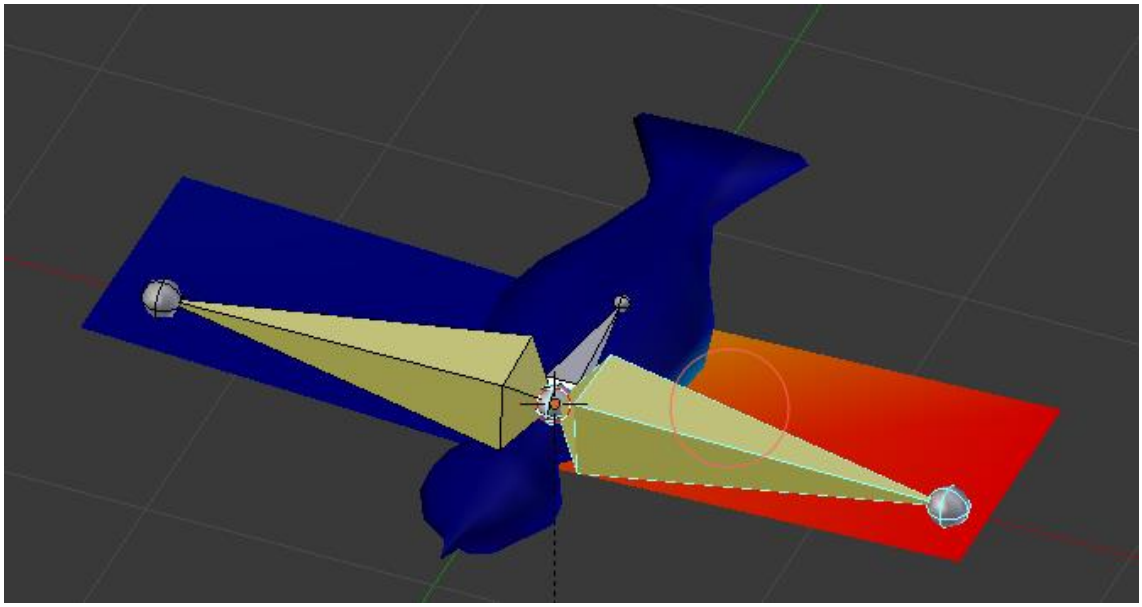

Figura 113: Pintando en *Weigh Peint* la parte de la malla a la que afecta el movimiento del ala izquierda

Se quiere mover de igual manera ambas alas. Para ello, primero se añaden objetos *Empty – Sphere* sobre cada una de las esferas de la armadura que hay en las alas. A estas nuevas esferas las llamo *Target Right* y *Target Left*.

Después se crea un cubo pequeño delante del pájaro. Cuando se mueva éste, las alas se moverán a su vez. Primero se seleccionan las dos esferas *Target* (hijos) y después el cubo (padre), Ctrl + P y se selecciona *Object (keep transform)*. Así, el cubo es padre de las dos esferas, que siguen su movimiento si se desplaza.

Ahora se debe hacer que la armadura siga el movimiento de las esferas *Target Left* y *Target Right*. Seleccionando cada uno de los huesos de las alas, en *Propierties – Bone constraints* y se añade *Inverse Kinematics:* se selecciona como objeto *Target* la esfera correspondiente, y en *Chain Lenght* se le da el valor 1, para que utilice únicamente un hueso.

Tras esto, al mover el cubo, las alas del pájaro se mueven simultáneamente.

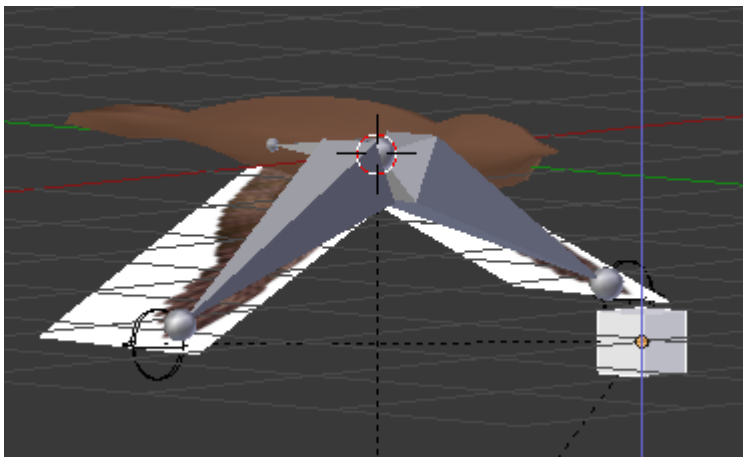

Figura 114: Movimiento de las alas al mover el cubo en el eje *z*

Se cambia *Screen Layout* de *Default* a *Animation*. Situándonos en el primer frame, se desplaza hacia abajo el cubo para que el pájaro comience con las alas abajo. Seleccionando el cubo, con I – *Location* se guarda la posición. Sin mover el cubo, se vuelve a guardar la posición en otro frame más adelante (en este caso, el 14). Después, en un frame intermedio (el 7), se mueve el cubo arriba y se guarda la posición del pájaro con las alas en alto. En la ventana *Dope Sheet* aparecen los *keyframes* marcados.

Tras esto se tiene una animación en la que el pájaro sube las alas y las vuelve a bajar. Para que esto se haga de forma repetida se va a *Graph Editor*, *Channel – Extrapolation Mode – Make Cyclic*.

Se añaden un par de pájaros más seleccionando el grupo de objetos actual y duplicándolo con Shift + D. Los nuevos pájaros se deforman al moverse al tener dependencia con el primero. Se soluciona añadiendo un objeto (por ejemplo, una *Ico Sphere*) nuevo para que sea el padre del resto de elementos. Como la armadura y el cubo

no dependen de nada, se seleccionan ambos, después el nuevo objeto y con Ctrl + P – *Object (Keep transform)* se hace *Ico Sphere* padre del resto. Si se duplica ahora, cada pájaro tiene su propio movimiento independiente de los demás.

En *Dope Sheet* aparecen los tres *keyframes* guardados para cada pájaro. Se seleccionan los de algunos de ellos, y con la tecla S puedo juntar o separar más los *keyframes* para hacer su movimiento más rápido o más lento, y que así no se muevan todos los pájaros exactamente igual.

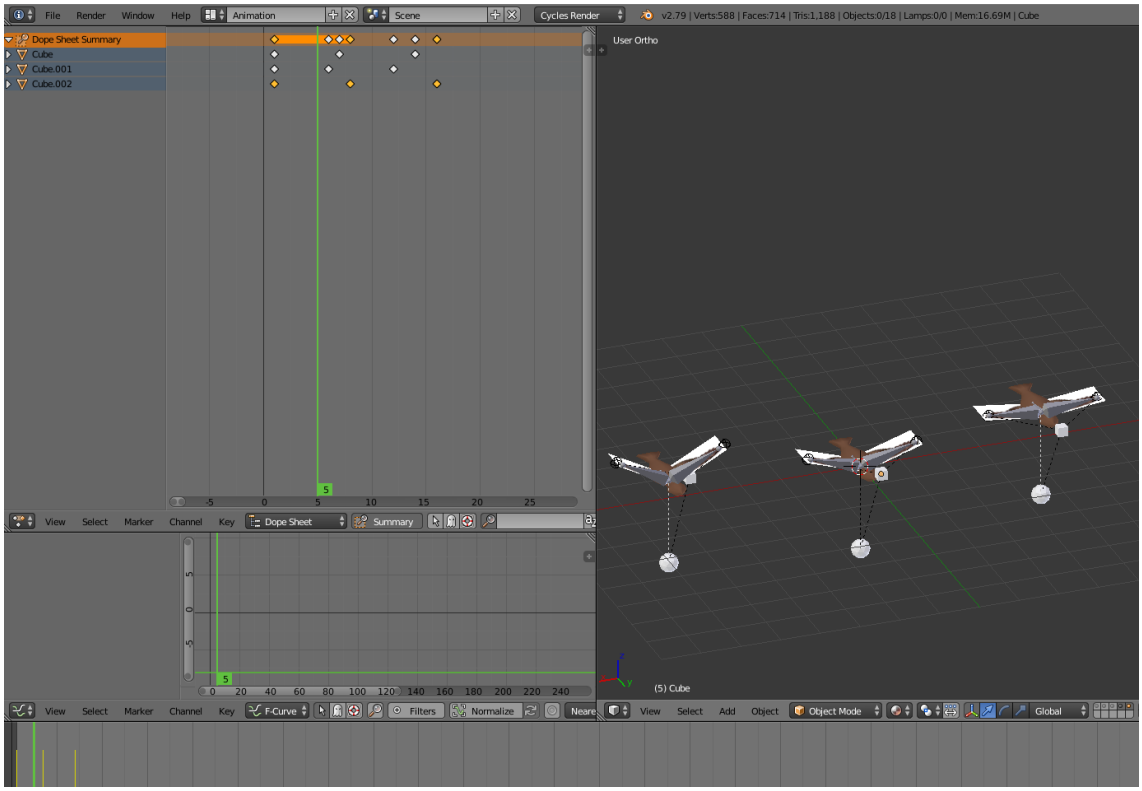

Figura 115: Animación del movimiento de alas de tres pájaros

Se seleccionan las mallas de todos los pájaros y se crea un grupo con ellas mediante Ctrl + G. El objetivo era que hicieran un recorrido mediante un sistema de partículas.

En primer lugar, se vuelve al *Screen Layout – Default* y se crea un objeto nuevo, en el cual se colocará el sistema de partículas: se crea una *Ico Sphere* y para poder ver en su interior, en las propiedades del objeto, apartado *Display – Maximum Draw Type* se selecciona *Wire*.

Se crea en ella el sistema de partículas. Se escoge el tipo *Emitter* y un número alto en *Lifetime* para que sean visibles durante mucho tiempo. En *Physics* se selecciona *Boids*, ya que este tipo de sistemas de partículas es controlado por inteligencia artificial y permite que las partículas sigan algunas reglas y comportamientos.

En el subapartado *Render* se desactiva *Emitter*, y así, con *Viewport* en *Render*, la esfera no se visualizará. Se selecciona *Group* y se escoge el grupo previamente creado. Se activa *Rotation* y *Scale* para que utilice las de los pájaros, y *Pick Random* para que vaya cogiendo del grupo cualquiera de los pájaros, de manera aleatoria. También se da algo de valor a *Random size* para que no tengan todos el mismo tamaño.

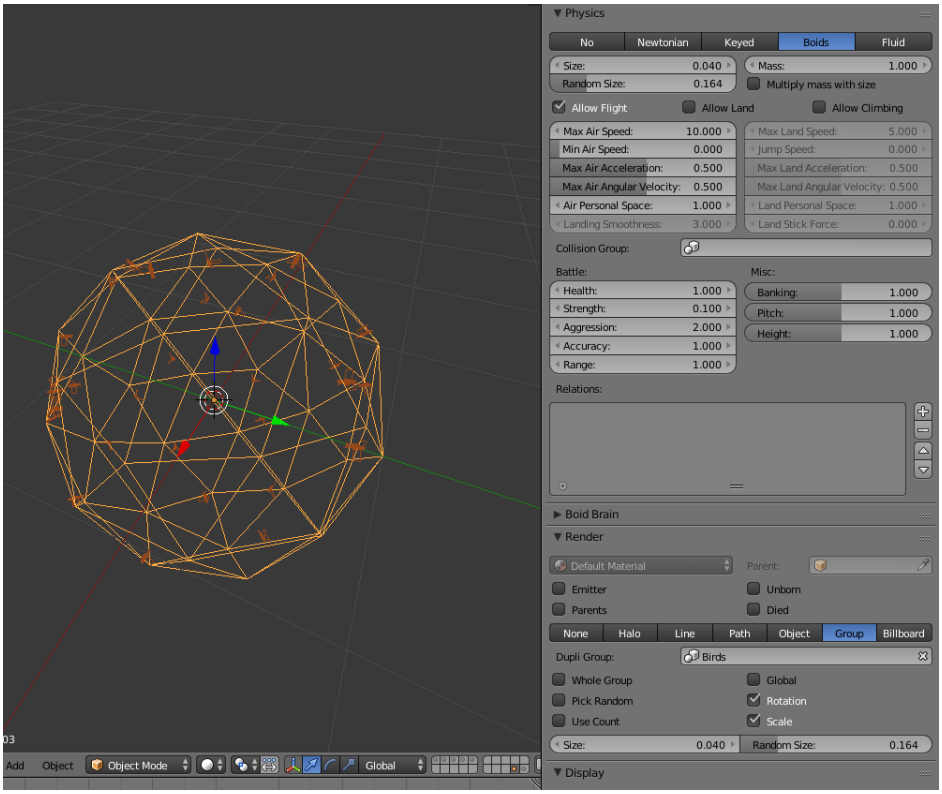

Figura 116: Sistema de partículas *Boids* en una *Ico Sphere*

Pulsando *Play* los pájaros se mueven por la esfera, sin chocar unos con otros. Esto es debido a las reglas de comportamiento que hay por defecto para *Boids*. En *Boid Brain*, se pueden editar y ordenar según la prioridad.

Se añade *Follow Leader*, que hará que los pájaros sigan a uno en concreto. Se crea un cubo con *Display – Maximum Draw Type* en *Wire*, al igual que antes, y en *Cycles Settings* se desactiva todo para que no sea visible con el *Viewport* en *Render*. El grupo de partículas se dirigirá hacia este objeto. Se añade la regla *Goal* seleccionando como objetivo al nuevo cubo. Ahora los pájaros salen de la esfera y se mueven hacia el cubo.

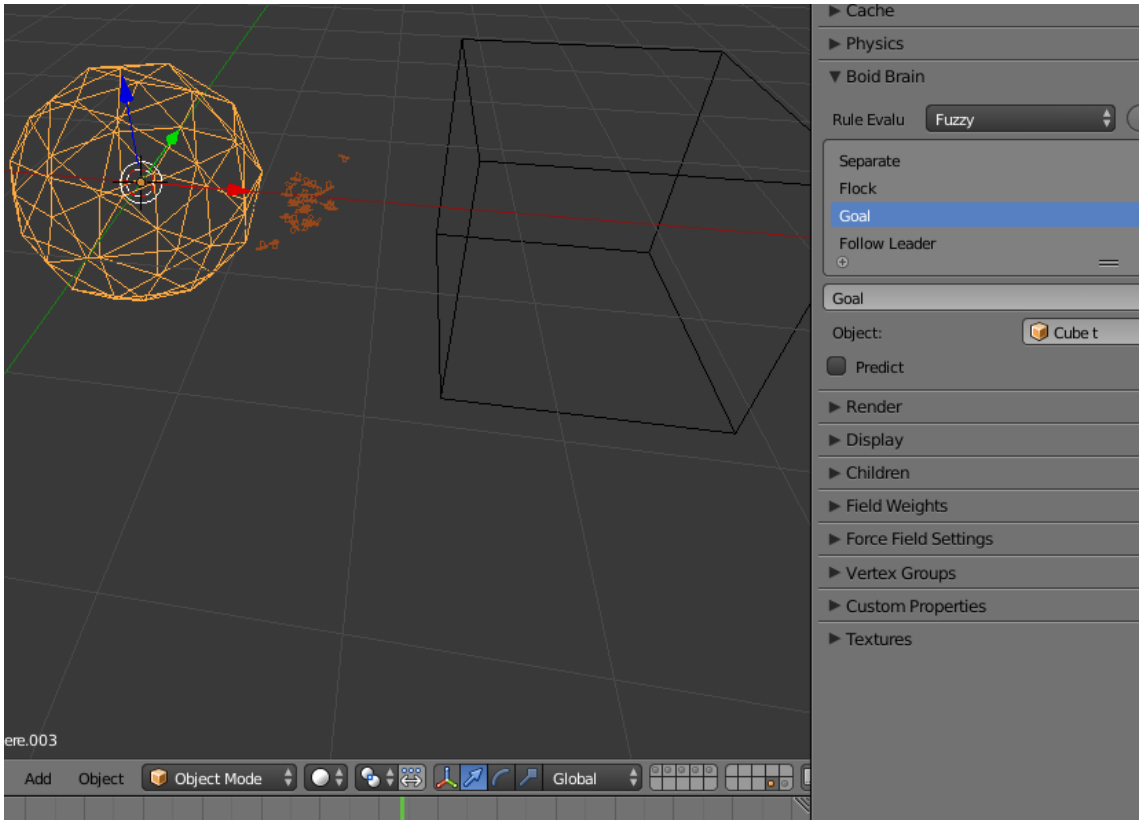

Figura 117: Partículas de pájaros dirigiéndose al cubo

Para que éste cubo no esté fijo en un mismo lugar, se hace que siga un camino. Con Shift + A – *Curve* – *Circle* se crea una curva. En *Edit Mode*, con *W* – *Subdivide* se añaden más segmentos a la curva y se desplazan para modificarla y que así no sea un círculo. Una vez terminada, en *Object Mode* se hace Ctrl + A – *Rotation & Scale* para aplicar la transformación.

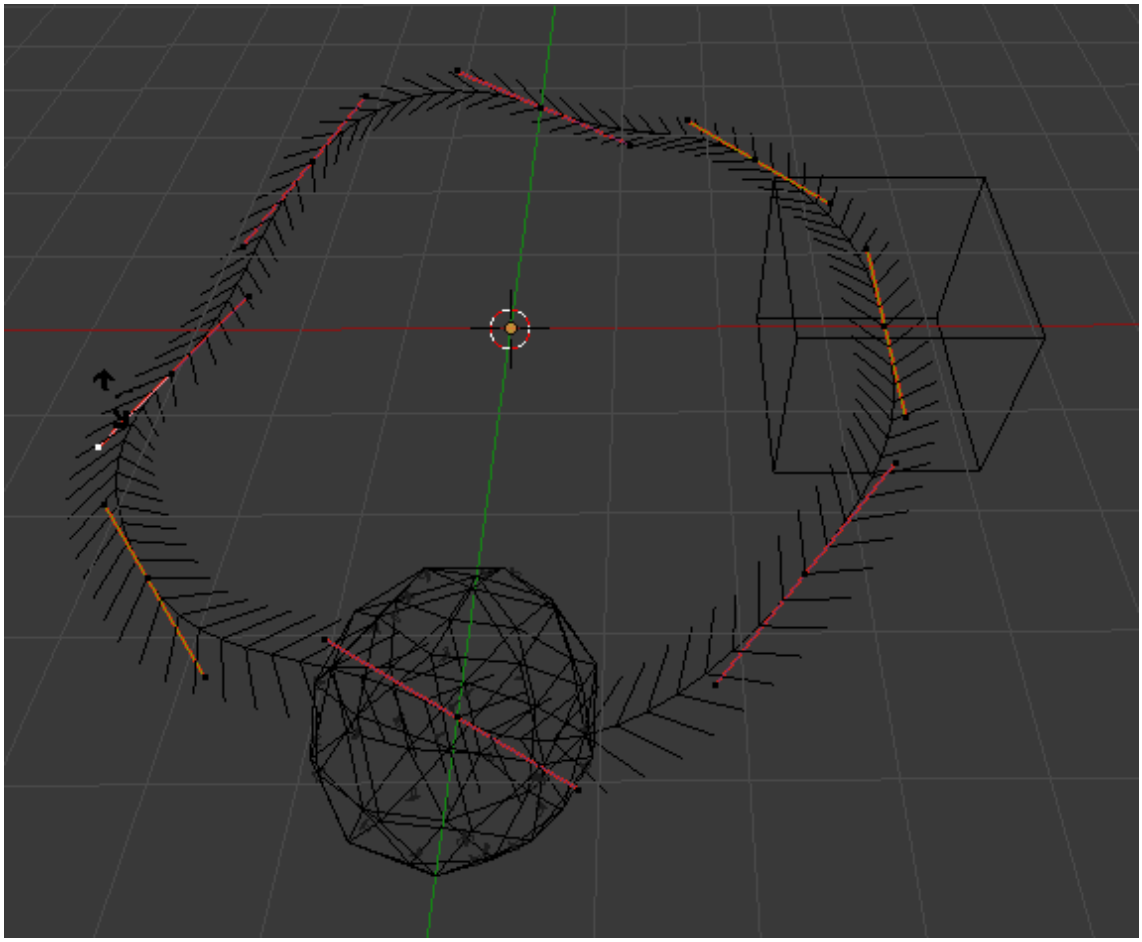

Figura 118: Haciendo camino a partir de una curva

Se selecciona el cubo, y en sus propiedades, *Constraints*. Se añade *Follow Path*: como objetivo se pone la curva, se activa *Follow Curve* y se hace click en *Animate Path*.

Con la curva seleccionada, en la propiedad *Data – Path Animation* se pone el mismo número de frames que se está utilizando en la ventana *TimeLine:* 500 frames.

Ahora, pulsando *Play*, los pájaros se van moviendo hacia donde esté el cubo, que a su vez sigue la curva. Se escala, haciéndola más pequeña, para que el cubo no se aleje tanto del grupo de pájaros.

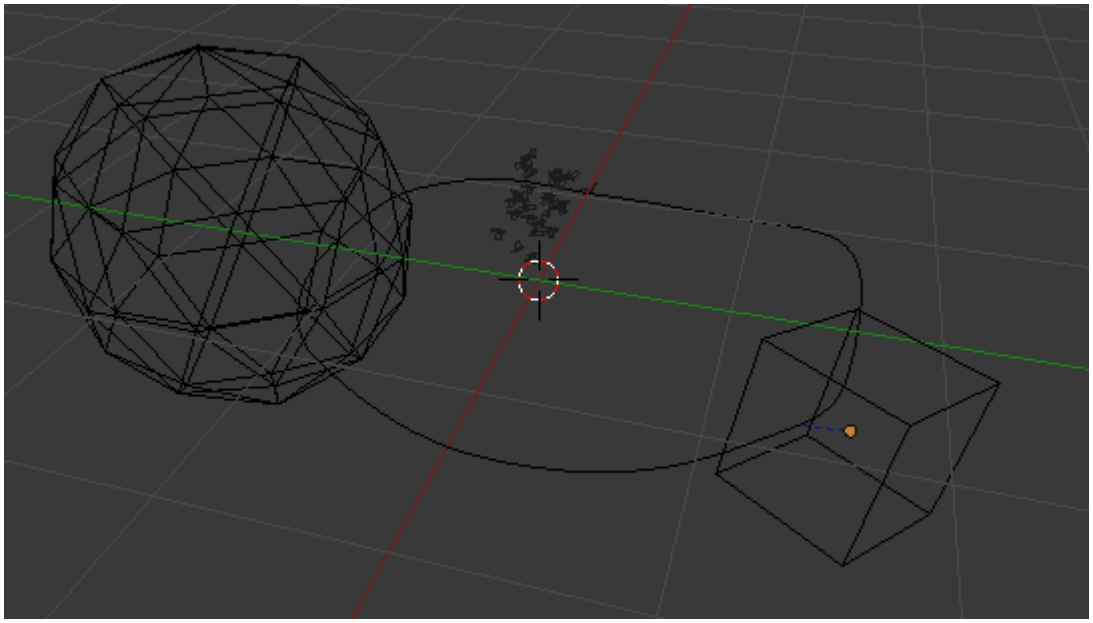

Figura 119: Animación de los pájaros siguiendo a un cubo que se mueve por una curva

Si se pone *Viewport* en *Render* y pulsando *Play* el grupo de pájaros comienza a moverse.

Figura 120: Grupo de pájaros en movimiento, visto en *Render Viewport*

No se puede exportar el sistema de partículas, por lo que se transforman las partículas a mallas con *Make duplicates real* o bien en *Modifiers* seleccionando *Convert* en el sistema de partículas. El problema es que esto provoca errores en las mallas a causa del esqueleto.

Aparte de esto, es preferible tener en la escena de Unreal un sistema de partículas en lugar de tantas mallas durante todo el tiempo.

Por esto, se decide modelar el pájaro en Blender y realizar la animación en Unreal.

Segundo pájaro

Modelado

Se realiza de manera similar al anterior, pero con la diferencia de que las alas fueron modeladas junto al cuerpo, en lugar de utilizar planos.

Se mantiene un modelo simple, sin muchos polígonos, puesto que los pájaros van a ser vistos siempre de lejos.

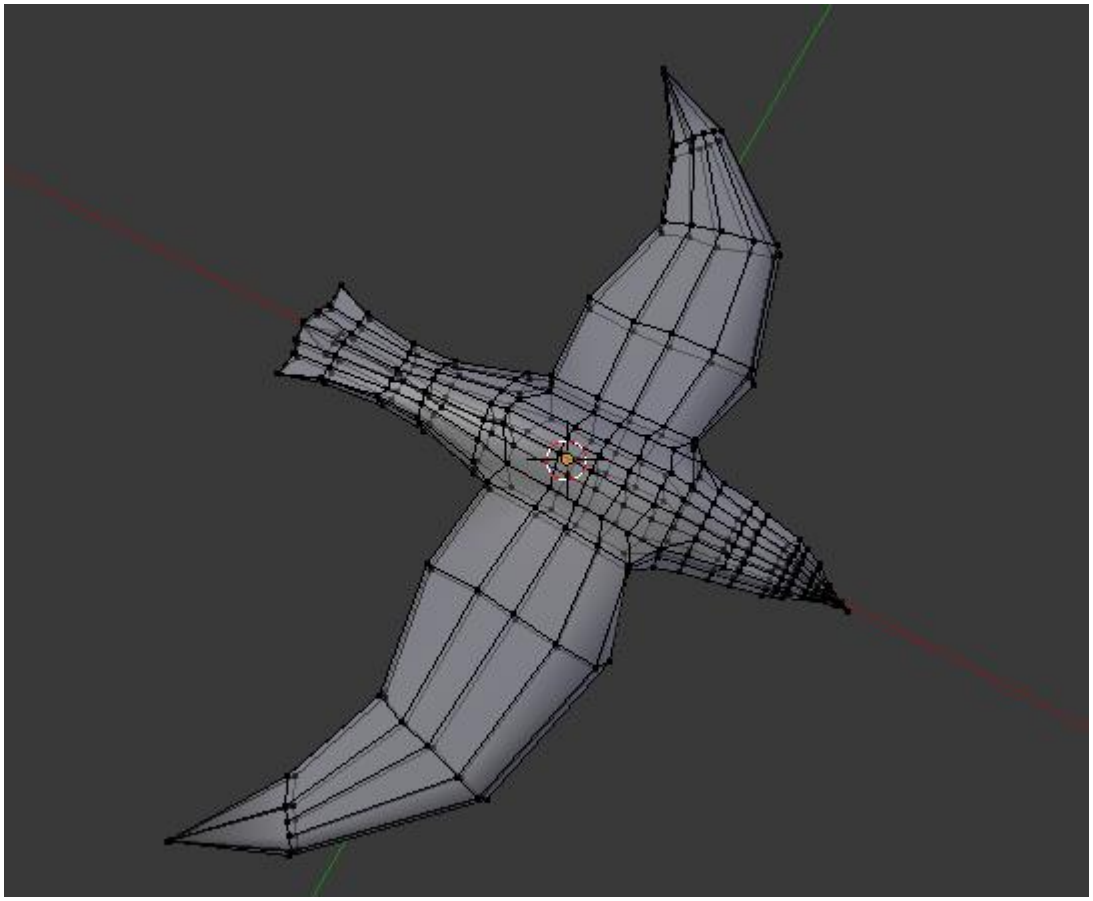

Figura 121: Nuevo modelo de pájaro, visto en *Edit Mode*

Texturizado y animación

En Unreal debía llevarse a cabo el movimiento de las alas, así como el de desplazamiento. Al contrario que en Blender, aquí los sistemas de partículas no pueden utilizar mallas animadas como partículas. Sí pueden ser mallas estáticas.

Se toma como ejemplo el proyecto gratuito Epic Zen Garden de la biblioteca de Epic Games, que posee varios sistemas de partículas de grupos de pájaros volando. Se utilizan mallas estáticas como partículas. El movimiento de las alas reside en el material y en los *Vertex Colors* de la malla.

En Blender, con el modo *Vertex Paint* se pinta una de las alas con color *RGB: 0, 1, 1* y la cola con *RGB: 1, 0, 1*. Con los nodos *Vertex Color* del material se moverán las diferentes partes de la malla.

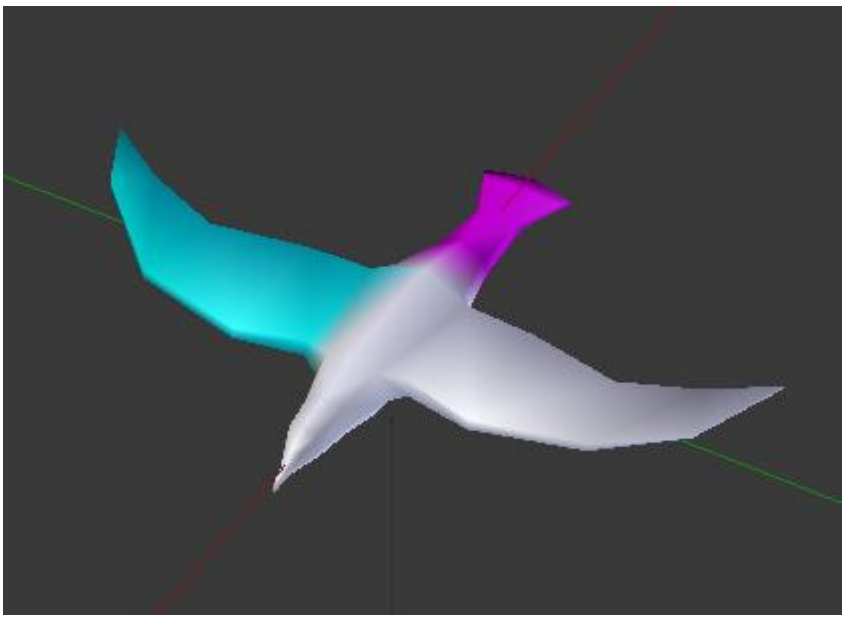

Figura 122: Pájaro pintado en *Vertex Paint*

Una vez importado a Unreal, entrando en la malla y seleccionando *Vert Colors* podemos ver estos colores.

Se utiliza el material incluido en el proyecto de Epic Zen Garden ya que proporciona todo lo necesario para el movimiento deseado del pájaro.

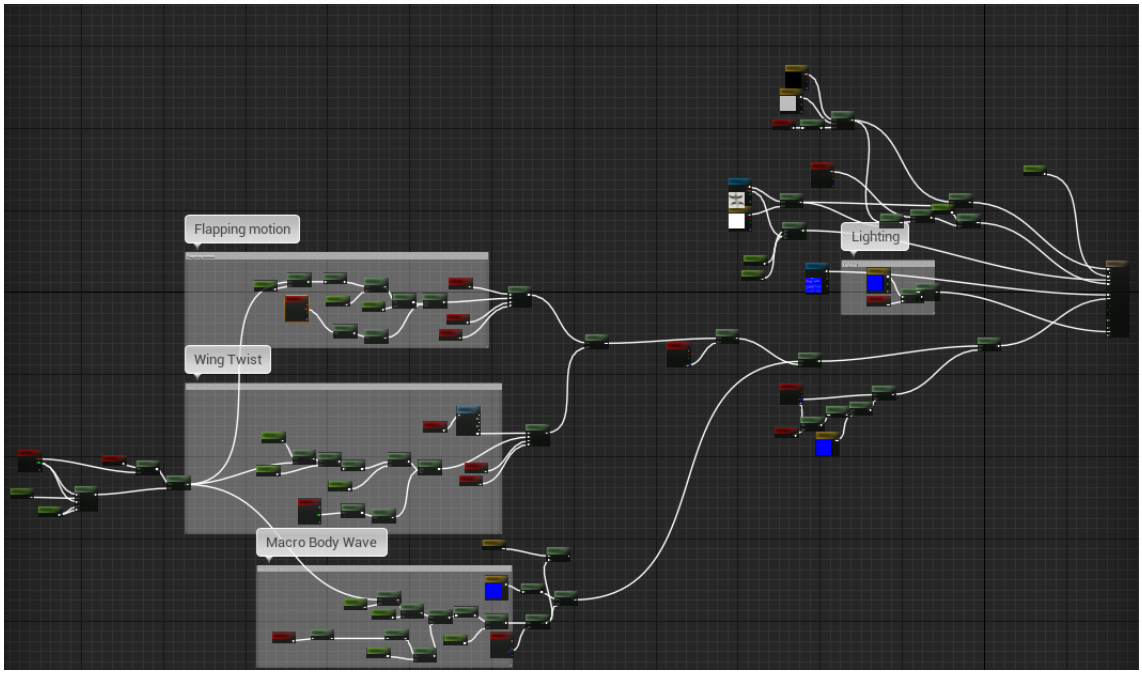

Figura 123: Material de los pájaros del proyecto Epic Zen Garden

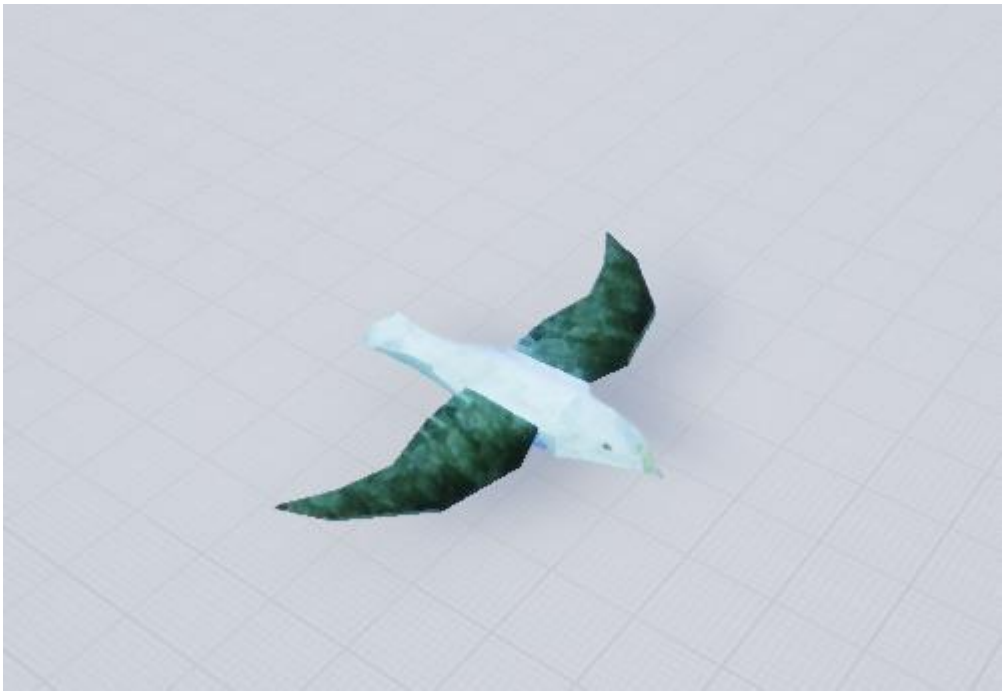

Figura 124: Pájaro con el material de Epic Zen Garden

5.5. Niveles de detalle

Unreal permite la tanto la importación de niveles de detalle o *LODs (Level Of Detail)*, así como la creación automática. Para la mayoría de los objetos se ha utilizado esta última opción.

En un principio sólo existe un nivel de detalle (*LOD base*), es decir, la malla tal cual se ha importado. Este *LOD* será el que se vea cuando al situar la cámara cerca de la malla, el que tiene más triángulos (y por tanto más detalle).

Se puede ver el *LOD* que se está visualizando, así como su número de triángulos y vértices y el tamaño de pantalla.

Figura 125: *LOD Base* de una roca

En el panel de *Ajustes de nivel de detalle (LOD)* hay varios grupos de niveles de detalle con unos valores por defecto. Como es preferible escogerlos manualmente, no se usa ningún grupo. Se pone la cantidad de *LODs* que se desee para la malla y se desactiva la casilla de *Calcular automáticamente las distancias del nivel de detalle*, ya que también se pondrán manualmente. En la figura 126, para una de las rocas, hay cinco niveles de detalle.

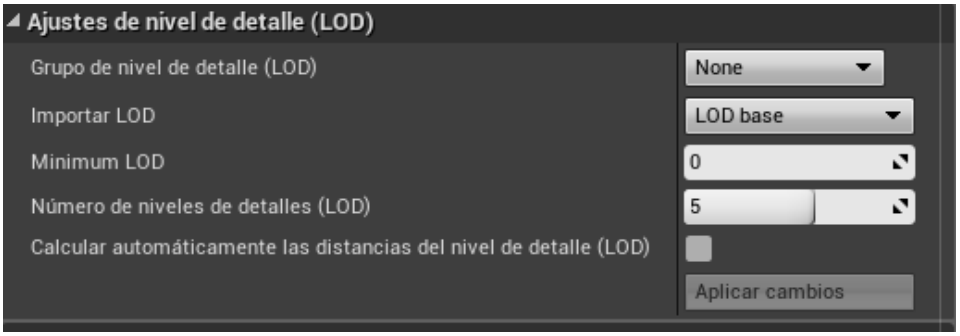

Figura 126: Ajustes de nivel de detalle de una roca

Una vez aplicados los cambios, se podrá poner la vista en cada uno de los niveles de detalle independientemente del tamaño de pantalla.

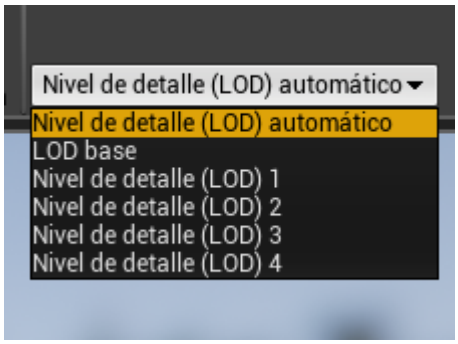

Figura 127: Selección de niveles de detalle para una roca con 5 *LODs*

Para cada nivel de detalle añadido, se escoge el valor del tamaño de pantalla a partir del cual, al alejarnos, la malla usará ese *LOD*, así como el porcentaje de triángulos del mismo respecto a la malla real. Automáticamente se recreará la malla con esa cantidad de triángulos. Se ajustan estos dos valores para que el cambio de un *LOD* a otro sea mínimamente visible. Esto se puede comprobar poniendo la malla en *Nivel de detalle LOD automático*, que según el tamaño de pantalla pondrá un *LOD* u otro.

Figura 128: Opciones del *LOD 4* de una roca

En la figura anterior, cuando el tamaño de pantalla sea 0.05, la malla tendrá el 5% de triángulos respecto a la original.

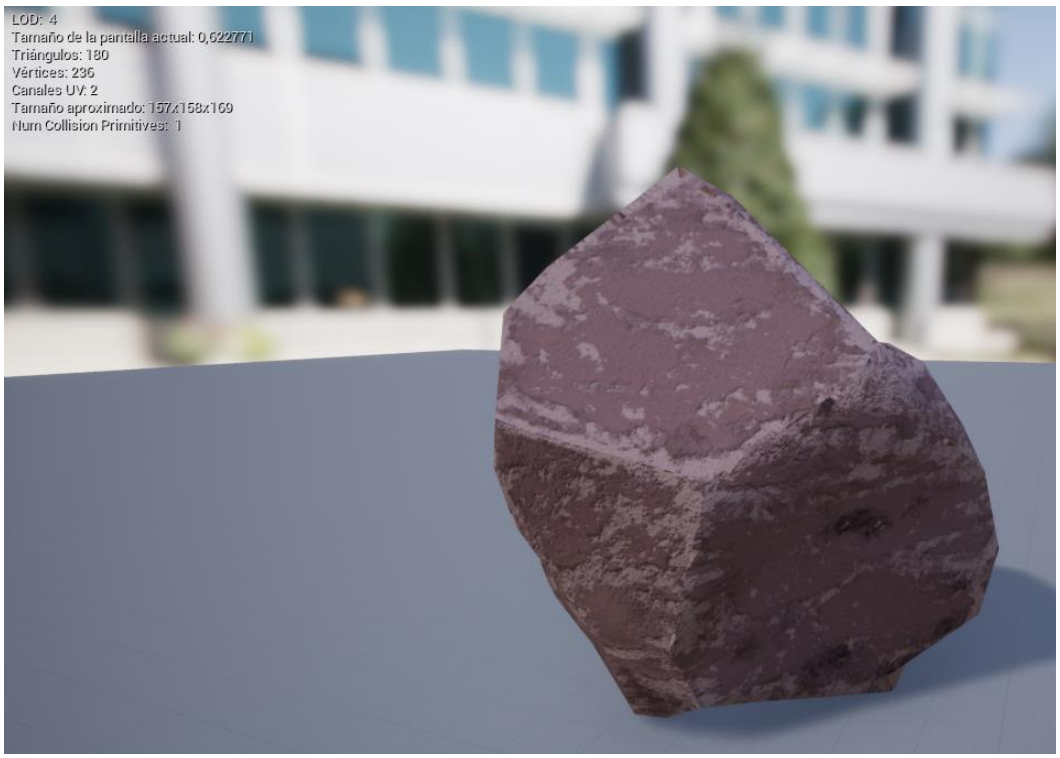

Figura 129: Vista del *LOD 4* de una roca

Como en *LOD automático* esta malla se verá mucho más pequeña, no se apreciará la reducción de polígonos.

Figura 130: Vista del *LOD automático* de una roca, alejada hasta utilizar el *LOD 4*

La malla de la figura 129 y 130 es la misma: *LOD 4*, vista de cerca (seleccionando el nivel de detalle correspondiente) y de lejos (seleccionando *LOD automático*).

Para algunos elementos, se han creado uno o más niveles de detalle manualmente, es decir, modelando.

Un ejemplo de esto son los árboles y las plantas: para los árboles, el último nivel de detalle consiste en dos planos que se cortan, ambos con la imagen del árbol completo como textura. En cuanto a las plantas, se han creado las mallas de nivel de detalle eliminando vértices de las mallas originales.

Figura 131: Último nivel de detalle de un árbol

5.6. Sistemas de partículas

5.6.1. Hojas cayendo

Se trata de un sistema de partículas que crea el efecto de hojas cayendo de un árbol. Se crea un objeto de tipo *Particle System*.

Haciendo doble click en él se abre el editor de sistemas de partículas de Unreal, *Cascade*.

Como el efecto a crear sólo tiene un elemento, las hojas, se necesita un único emisor o *emitter*. Dentro de éste se editan diferentes *modules*; los componentes del *emmiter* que definen aspectos concretos de comportamiento:

Necesario

Se añade el material a utilizar. Esto es, la instancia de material de la hoja.

Spawn

Se determina la cantidad de partículas que se va generando. Se pone un valor bajo para que no caigan muchas hojas de seguido.

Lifetime

Se tiene la duración de existencia de cada partícula. Una vez acabado el sistema, colocando éste sobre un árbol a la altura deseada, se determina cuánto tiempo debe durar la partícula hasta desaparecer en el suelo.

Initial Size

Se elige el tamaño que tendrán las partículas, que puede variar entre un tamaño mínimo y uno máximo.

Velocidad Inicial

La velocidad en los ejes x e y es positiva, mientras que en el eje z es negativa, puesto que las hojas van hacia el suelo. Se pone también velocidad en x e y para que no caigan en una línea perpendicular al suelo.

Ubicación inicial

Con los valores máximos y mínimos de cada eje se añade un rango en el cual las hojas van apareciendo, de modo que no salgan todas del mismo punto. Aparecen en diferentes alturas y por diferentes zonas del árbol.

Actor collision

Se elige la opción de *matar* a la partícula, para que una vez llegue al suelo no siga existiendo.

Órbita

Las partículas rotan en su posición actual. Esto hace que no caigan uniformemente describiendo una línea recta. Así, las hojas parecen ser mecidas por el viento conforme caen.

Rotación inicial

La rotación de la hoja cuando aparece. Se añade una rotación mínima y máxima, para que no aparezcan todas con la misma.

Proporción de la rotación inicial

Rotación de las hojas sobre sí mismas.

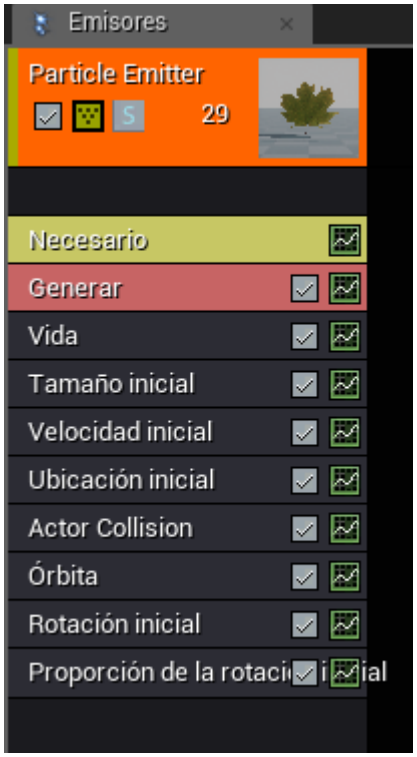

Figura 132: Emisor y módulos del sistema de partículas de las hojas cayendo

Con esto ya está realizado el sistema de partículas. Arrastrándolo a la escena, se ve lo siguiente:

Figura 133: Sistema de partículas de hojas cayendo en la escena

Para añadir el sistema de partículas a un árbol, se crea un *blueprint*. Aquí se arrastran el árbol y el sistema de partículas, y se sitúa éste en la posición deseada sobre el árbol.

Figura 134: Sistema de partículas de hojas cayendo y árbol en un *blueprint*

En la escena se arrastra este *blueprint* para tener el árbol con sus hojas cayendo al suelo.

Figura 135: *Blueprint* del árbol con las hojas cayendo, colocado en la escena

5.6.2. Pájaros volando

Se crea un *Particle System*. Como partícula se va a utilizar una malla, por lo que se añade el módulo *Datos de la malla* y se arrastra la malla del pájaro.

A partir de uno los sistemas de partículas del proyecto Epic Zen Garden, se cambian de los parámetros de los módulos hasta conseguir el efecto deseado.

5.7. Entorno 3D

Sobre el suelo del entorno hay elementos pequeños, que se aprecian de cerca: hojas, flores, ramitas, piedras, hierbajos y algunas plantas. Por otra parte, desde más lejos se ven las rocas, los árboles, y el agua. De algunos árboles caen hojas, y hay pájaros sobrevolando el cielo.

5.7.1. Colocación de elementos en la escena. Modo *Follaje*

Algunos objetos, como las montañas, son colocados manualmente en la escena. Sin embargo, para aquellos que aparecen una gran cantidad de veces se utiliza la herramienta *Follage*, que permite colocar múltiples instancias de uno o más objetos.

Se arrastran aquí las mallas de árboles, piedras, flores, plantas, etc, lo cual genera objetos *Foliage Type*. Se pueden activar o desactivar, por lo que es posible colocar varios tipos de malla de forma grupal. Por ejemplo, para poner flores se seleccionan los diferentes tipos de flor para que se coloquen a la vez, mezclándose unas con otras.

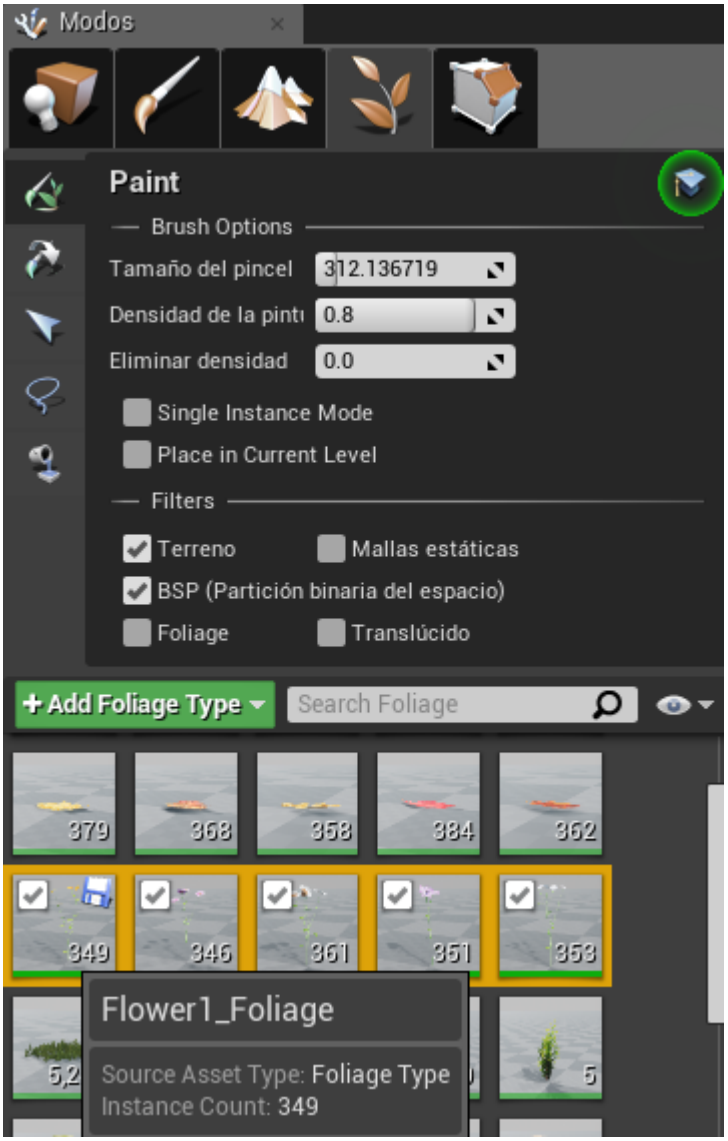

Figura 136: Modo Follaje

Sobre la escena aparece una herramienta de pincel. Se ajustan las opciones de tamaño o densidad para determinar la cantidad de elementos a colocar y lo juntos que están los unos de los otros.

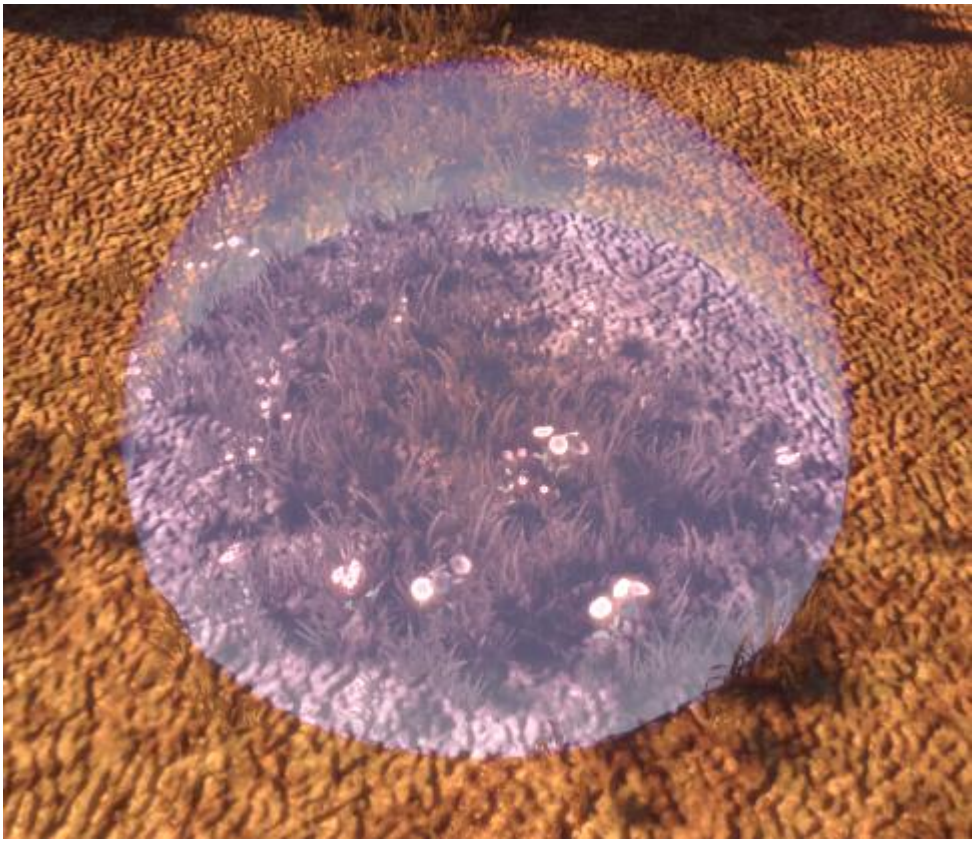

Figura 137: Pincel del modo *Follaje* colocando flores sobre el terreno

Adicionalmente, para cada uno de los *Foliage Type* se puede editar su propia densidad, entre otros parámetros.

Se escoge una escala mínima y máxima para que las instancias de cada objeto tengan diferentes alturas.

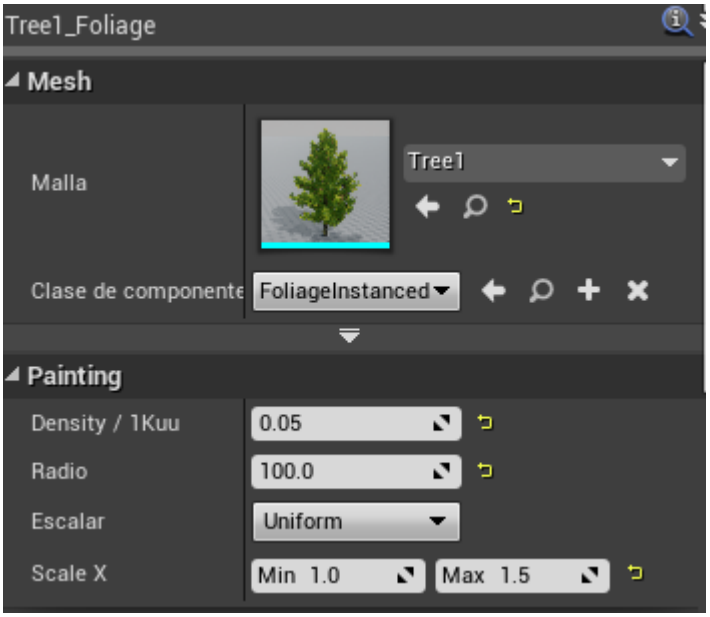

Figura 138: Parte del panel de detalles de un *Foliage Type*

5.7.2. Sonidos

Se ha introducido un audio de pájaros trinando descargado de *freesound.org* para utilizar como sonido ambiental.

Se crea a partir de un tipo de dato de Unreal que posee un editor de nodos de sonido: *Sound Cue* o pista de sonido.

En primer lugar, se importa el archivo. Se ha utilizado como formato el *.wav*. Éste se arrastra en la pista de sonido creada. Se enlaza con un nodo *Modulator*. Este nodo permite escoger el ángulo (que interfiere en el tono, es decir, los agudos y graves) y volumen máximos y mínimos, lo cual añade variación en el sonido. El resultado se enlaza con el nodo de salida.

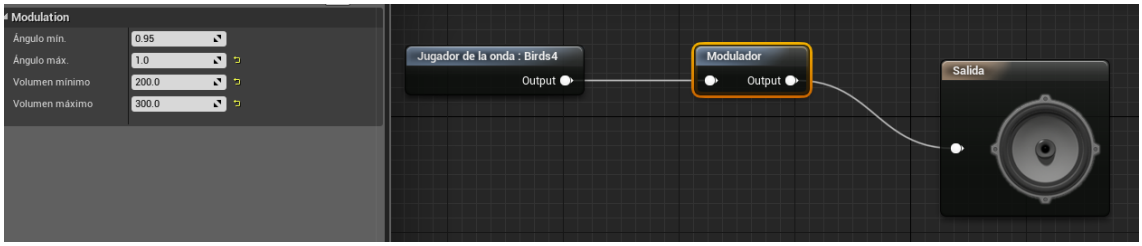

Figura 139: Nodos en la pista de sonido de pájaros trinando y parámetros del nodo *Modulator*

Como se desea este sonido por toda la escena no se le añade atenuación. La variedad generada con *Modulator* es suficiente para poder atravesar todo el entorno sin apreciar uniformidad o repetición en el sonido.

5.7.3. Iluminación y postprocesado

Se utilizan varios elementos para el iluminado de la escena:

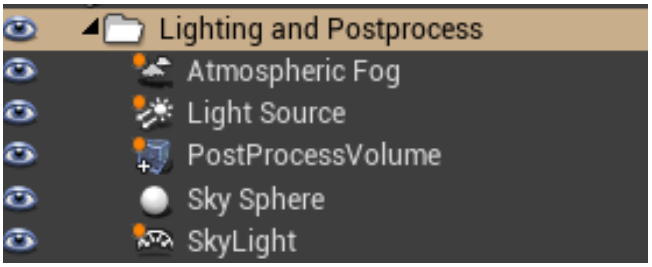

Figura 140: Elementos de iluminación y postprocesado utilizados en el entorno

En primer lugar, se crea una fuente de luz direccional (*Light Source*). Se sitúa a cierta altura sobre el entorno y se escoge la movilidad *Stationary*. Se ajusta el color de la luz, la intensidad, el brillo, la temperatura, etc, hasta conseguir un efecto de luz otoñal.

Las sombras no permiten ver nada de lo que hay en ellas. Por esto, se añade *SkyLight* y se sitúa sobre el suelo. Las figuras 141 y 142 muestran la misma sombra sin y con *SkyLight*, respectivamente.

Figura 141: Sombra cuando no hay *SkyLught*

Figura 142: Sombra cuando hay *Skylight*

Con *Sky Sphere* se cambia el color del cielo. También se añaden nubes, cuya velocidad de movimiento puede ajustarse, así como el color y la opacidad.

Se añade *PostProcessVolume* para ajustar el color general de la escena (se activa *Infinite Extent* para que la abarque completamente). Además permite desenfocar tanto los elementos lejanos a la cámara como los muy cercanos: se utiliza para "alejar" las montañas y cuando la cámara pasa muy cerca de algo.

En las figuras 143 y 144 se puede apreciar la diferencia entre poner desenfoque o no, fijando la vista en las montañas y en las hojas del árbol en el que se sitúa la cámara.

Figura 143: Parte de la escena sin desenfoque

Figura 144: Parte de la escena con desenfoque de elementos muy cercanos y lejanos

Finalmente, con *Atmospheric Fog* se crea un efecto de luz penetrando en la atmósfera.

Figura 145: Cielo sin *Atmospheric Fog*

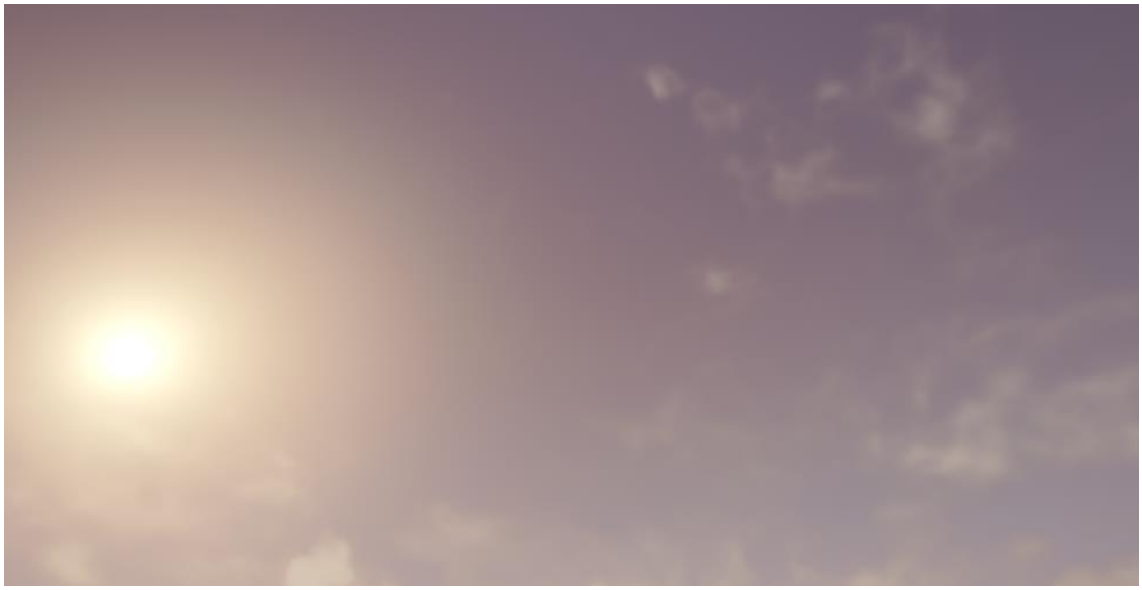

Figura 146: Cielo con *Atmospheric Fog*

5.8. Realidad virtual en la escena: Oculus Rift

Se han añadido tanto las gafas de realidad virtual Oculus Rift como sus mandos, Oculus Touch, al entorno.

Una vez instalados las gafas y los mandos, en Unreal se activa el plugin *OculusVR* y la escena se podrá reproducir en realidad virtual.

Figura 147: Probando Oculus Rift + Touch en una escena de prueba de Oculus (I)

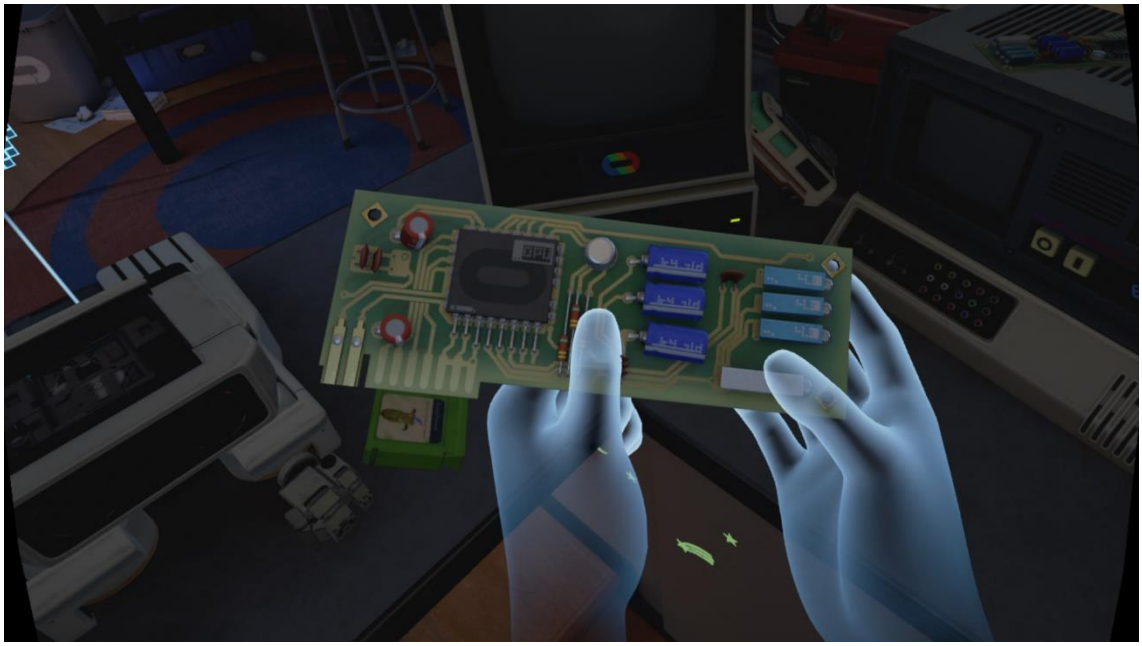

Figura 148: Probando Oculus Rift + Touch en una escena de prueba de Oculus (II)

Para ver en pantalla lo que se está viendo a través de cada una de las lentes de las gafas, una vez iniciada la reproducción de la escena en realidad virtual se abre la consola y se introduce el comando *hmd mirror mode 1*.

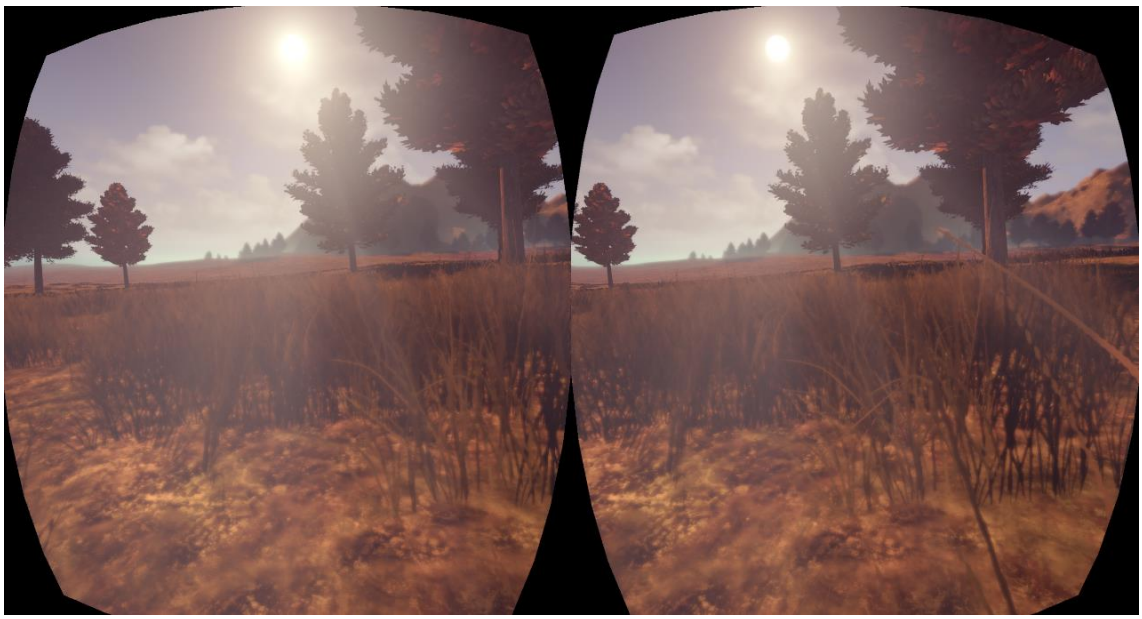

Figura 149: Entorno visto a través de Oculus Rift (I)

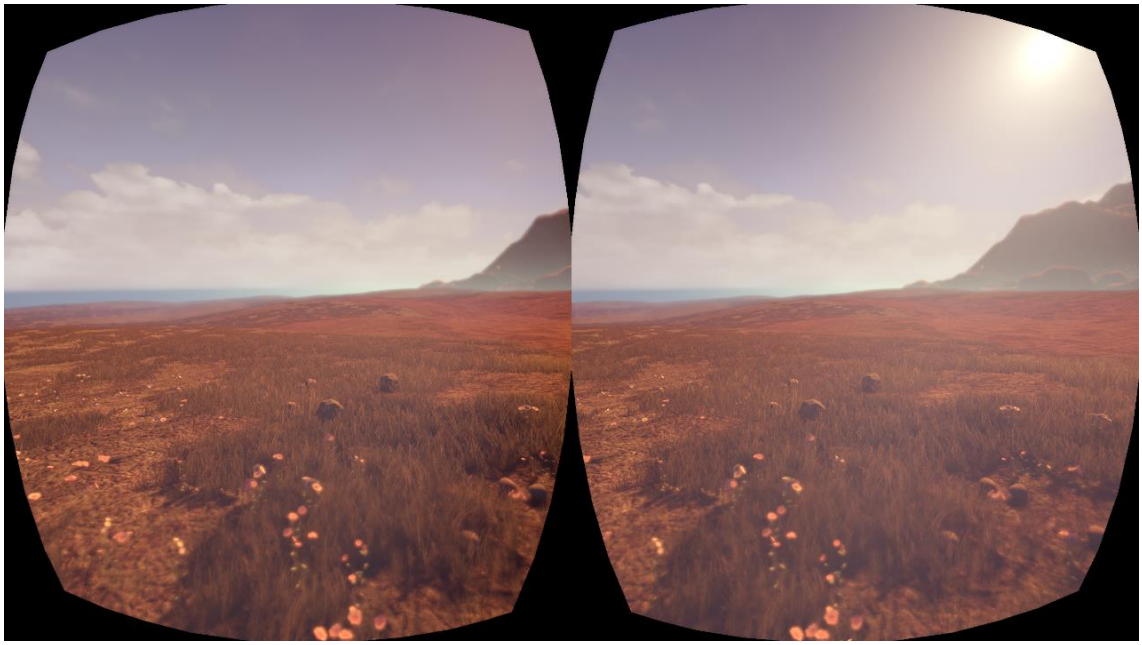

Figura 150: Entorno visto a través de Oculus Rift (II)

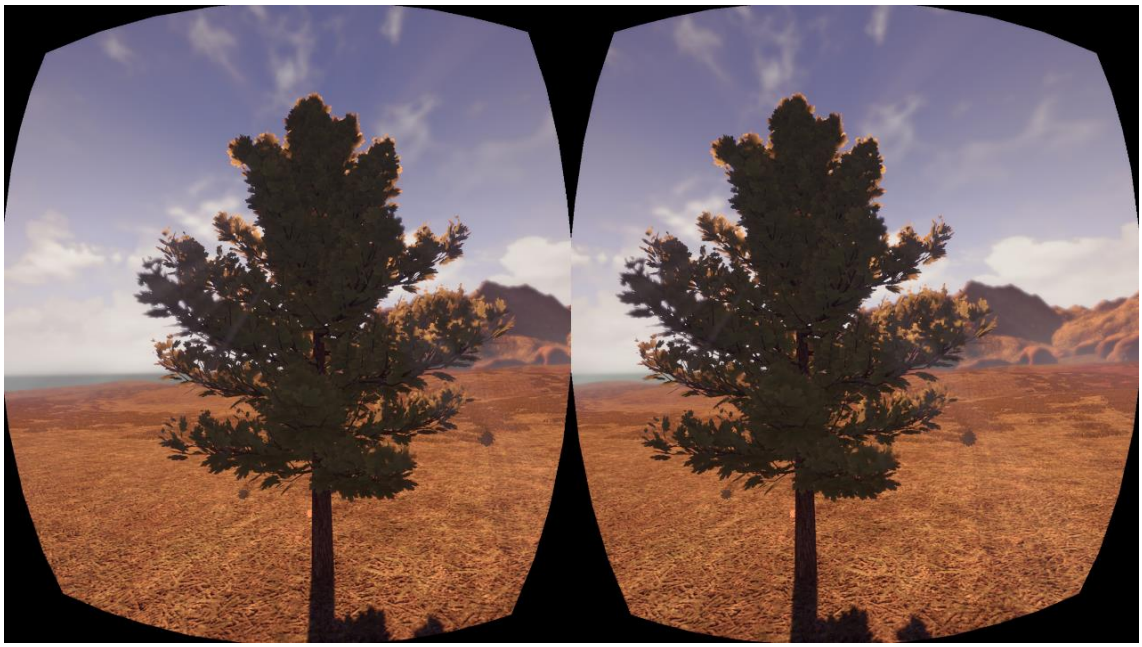

Figura 151: Árbol con las hojas cayendo, visto a través de Oculus Rift

A pesar de que no hay una interacción con los elementos de la escena, ésta se ha probado además con Oculus Touch.

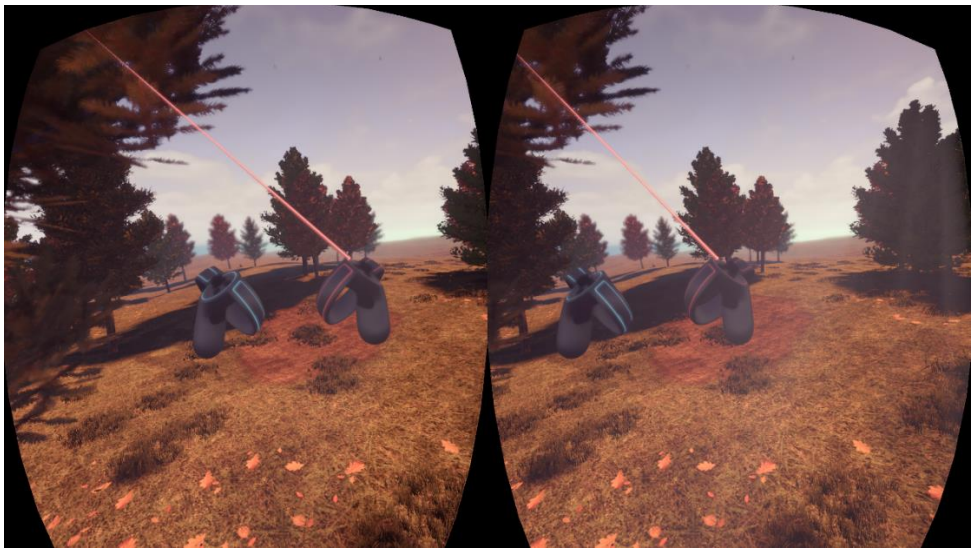

Figura 152: Entorno visto en realidad virtual con Oculus Touch añadidos

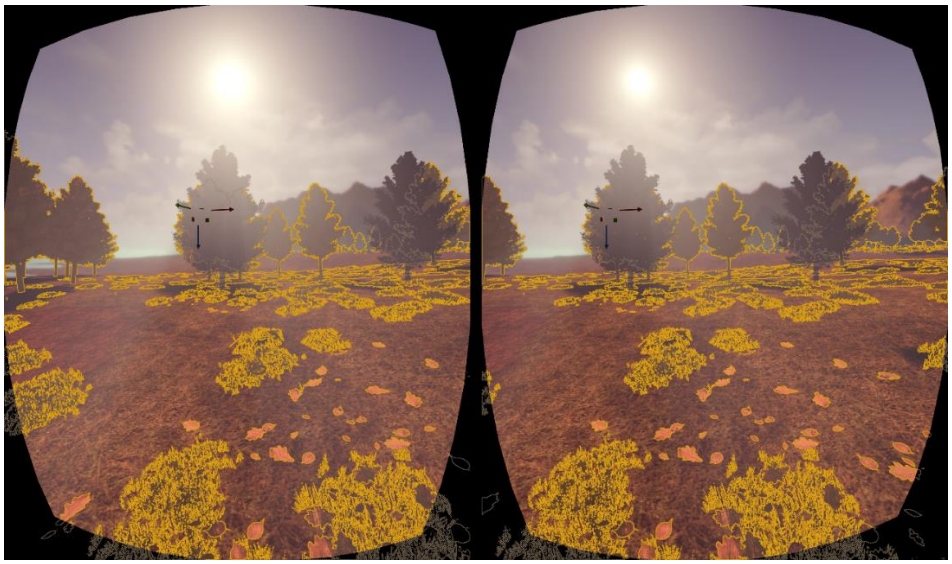

Figura 153: Editor de Unreal en realidad virtual; selección de elementos con los Oculus Touch

Conclusiones

Considero haber cumplido mis objetivos en la realización de este proyecto: el entorno realizado puede ser utilizado en un videojuego o una simulación, visualizándose tanto en el monitor de un ordenador como a través las gafas Oculus Rift.

Durante todo el desarrollo he aprendido mucho sobre realidad virtual, modelado, materiales y texturas, y, sobre todo, he aprendido a utilizar gran parte de las muchas características que ofrece Unreal Engine 4.

Por supuesto, han surgido errores –más de los explicados en esta memoria– y ha sido necesario repetir tareas ya realizadas para corregirlos: las cosas que conforman el proyecto final no se han obtenido la primera vez que se llevan a cabo, algunos ni a la segunda, ni a la tercera, etc. También ha habido otras cosas que no se han repetido debido a un error, sino porque no me convencía su resultado.

Finalmente, he aprendido a llevar a cabo un proyecto extenso, organizándolo mediante objetivos a corto plazo y priorizando tareas para lograr tener el producto terminado, con un gran margen de tiempo para mejorarlo.

Bibliografía y referencias

Recursos empleados

Algunos recursos se han adquirido a través de internet, teniendo éstos licencia gratuita de uso:

- Assets de hierba, rocas y otros elementos de la naturaleza:

  https://megascans.se

  Open World Demo Collection, de la biblioteca de Epic Games.

  https://forums.unrealengine.com/community/community-content-tools-and-tutorials/30694-free-foliage-starter-kit?59812-Free-Foliage-Starter-Kit

- Material y sistema de partículas de pájaro:
  Proyecto Epic Zen Garden, en la biblioteca de Epic Games.

- Planos de agua:
  Water Planes, en la biblioteca de Epic Games.

- Pinceles para esculpir en Blender:
  https://www.blendernation.com/2015/02/17/free-blender-brushes/
  https://www.blendernation.com/2017/08/26/free-download-18-rock-brushes

- Sonidos:
  https://freesound.org

- Texturas utilizadas:
  https://www.textures.com

Material consultado

Documentación

Blender 2.79: https://docs.blender.org/manual/en/dev/index.html

Oculus: https://developer.oculus.com/documentation/unreal/1.13/concepts/unreal-hmd/

Substance Painter: https://support.allegorithmic.com/documentation/display/SPDOC/Substance+Painter

Unreal Engine 4: https://docs.unrealengine.com/en-us/

World Machine: http://www.world-machine.com/learn.php

Libros

McCaffrey, M. (2017). *Unreal engine VR cookbook: Developing Virtual Reality with UE4*.

Parisi, T. (2015). *Learning Virtual Reality*. [s.l.]: O'Reilly Media.

Sherman, W. (2017). *Understanding Virtual Reality*. [s.l.]: Morgan Kaufmann.

Vídeos

Curso: *Learning Path: Virtual Reality Fundamentals* – O'Reilly.